Programmable graphics processor having pixel to character conversion hardware for use in a video game system or the like.

Patent Number:	EP0553531, A3
Publication date:	1993-08-04
Inventor(s):	GRAHAM CARL N (GB); CHEESE BEN (GB); SAN JEREMY E (GB); WARNES PETER R (GB)
Applicant(s):	AN INC (US)
Requested Patent:	CN1080200
Application Number:	EP19920307134 19920805
Priority Number(s):	US19920827201 19920130
IPC Classification:	G06F15/72
EC Classification:	G06T17/00
Equivalents:	AU2060592, AU654727, CA2074388, CN1048563B, JP6079063, KR270198
Cited Documents:	EP0402067; EP0431724; US5004232

Abstract

A fully programmable, graphics microprocessor is disclosed which is designed to be embodied in a removable external memory unit for connection with a host information processing system. In an exemplary embodiment, a video game system is described including a host video game system and a pluggable video game cartridge housing the graphics microprocessor. The game cartridge also includes a read-only program memory (ROM) and a random-access memory (RAM). The graphics coprocessor operates in conjunction with a three bus architecture embodied on the game cartridge. The graphics processor using this bus architecture may execute programs from either the program ROM, external RAM or its own internal cache RAM. The fully user programmable graphics coprocessor has an instruction set which is designed to efficiently implement arithmetic operations associated with 3-D graphics and, for example, includes special instructions executed by dedicated hardware for plotting individual pixels in the host video game system's character mapped display which, from the programmer's point of view, creates a "virtual" bit map by permitting the addressing of individual pixels -- even though the host system is character based. The graphics coprocessor interacts with the host coprocessor such that the graphics coprocessor's 16 general registers are accessible to the host processor at all times.

Data supplied from the **esp@cenet** database - I2

[19]中华人民共和国专利局

[11] 公开号 CN 1080200A

[12] 发明专利申请公开说明书

[21]申请号 92112791.X

[43]公开日 1994 年 1 月 5 日

[51]Int.Cl[5]
A63F 9/22

[22]申请日 92.10.31

[30]优先权

[32]92.1.30 [33]US[31]07/827,201

[71]申请人 A/N股份有限公司

地址 美国华盛顿州

[72]发明人 杰里米E·桑 本·奇斯

卡尔 N·格雷汉姆

彼特 R·沃内斯

[74]专利代理机构 上海专利事务所

代理人 顾承根

H04N 5/14 G11C 17/00

说明书页数: 84 附图页数: 24

[54]发明名称 图形处理器

[57]摘要

本发明公开了一种全编程图形微处理器，它配置在外部存储单元（例如插入式电视游戏卡）中，与配置于游戏卡上的三总线结构结合使用。该图形协处理器可执行来自程序 ROM，外部 RAM 或其本身内部的超高速缓存 RAM 的程序，具有实现有关 3-D 图形算术运算的指令集，例如包括由专用硬件执行将单个像素标绘在主系统字符映像显示中的指令，建立"虚拟的"位映像，并与主处理器交互作用使其 16 个通用寄存器都可由主处理器存取。

权 利 要 求 书

1. 一种信息处理系统(20,19)中的外部存储系统(19)，该信息处理系统与显示屏(36)一起使用，并具有执行电视图形程序的微处理器(22)和存储组合时确定一显示帧的多个字符的字符数据的图像存储器(30)，该外部存储系统(19)的特征在于它包括：

程序存储器(10)，可至少存储所述电视图形程序的某些指令；

变换电路(2,52)，与所述程序存储器相连，接收按照像素规格表示的显示数据以处理所述像素规格并且将所述像素规格数据变换成所述图像存储器(30)所采用的形式的字符数据。

2. 如权利要求 1 所述的外部存储系统，其特征在于所述像素规格包括在显示屏(36)上确定像素位置的坐标数据，所述变换电路(2,52)接收从所述程序存储器来的所述像素规格。

3. 如权利要求 2 所述的外部存储系统，其特征在于所述变换电路(52)包括地址变换器电路(202,204,HA,FA,214,216)以接收像素坐标数据并产生指定字符的地址。

4. 如权利要求 1 所述的外部存储系统，其特征在于还包括随机存取存储器(RAM)(6,8)，临时存储所述变换电路所产生的字符数据的缓冲存储器装置(206)，和将存储在所述缓冲存储器中的字符数据送给所述 RAM 的装置(206,228,88)。

5. 如权利要求 2 所述的外部存储系统，其特征在于还包括临时存储来自所述程序存储器的像素坐标数据的寄存器装置(202,204)。

6. 如权利要求 1 所述的外部存储系统，其特征在于还包括与所述程序存储器相连的可编程图形处理器(2)，而所述变换电路(52)配置在所述图形处理器中。

7. 如权利要求 6 所述的外部存储系统，其特征在于所述可编程

图形处理器包括第一源端公共总线(X),第二源端公共总线(Y)和目的端公共总线(Z),所述变换电路接收来自所述第一源端总线(X)和所述第二源端总线(Y)的数据并将数据送到所述目的端总线。

8.如权利要求1所述的外部存储系统,其特征在于所述像素规格包括像素的显示坐标和与所述显示坐标相关的彩色信息,还包括接收并临时存储所述彩色信息的彩色寄存器装置(54)。

9.如权利要求8所述的外部存储系统,其特征在于还包括接收从所述彩色寄存器装置来的像素彩色信息的寄存器矩阵(206)。

10. 如权利要求1所述的外部存储系统,其特征在于还包括随机存取存储器(RAM)(6,8)和控制对所述RAM存取的RAM控制器(88),所述变换电路包括:产生字符地址的地址变换装置(202,204,HA,FA,214,216),和产生包括与所述像素规格对应的数据在内的字符数据的字符数据产生装置(202,206),和将由所述变换电路所产生的字符地址和字符数据传送给所述RAM控制器(88)的装置(206,216)。

11. 如权利要求1所述的外部存储系统,其特征在于还包括随机存取存储器(RAM)(6,8),和将字符信息从所述RAM送给所述变换电路以便与由该变换电路处理的字符规格并联起来的装置(82,206)。

12. 如权利要求1所述的外部存储系统,其特征在于还包括存储字符数据的装置,所述变换电路(52)包括从所述存储装置接收目前处理的像素附近的待显示的其他像素的字符数据信息的装置。

13. 如权利要求1所述的外部存储系统,其特征在于程序存储器(10)和所述变换电路(2,52)配置在电视游戏卡(19)中。

14. 如权利要求1所述的外部存储系统,其特征在于所述变换电路包括存储与指定像素和包含有该指定像素的字符中的其他像素有关的数据的彩色矩阵(206)。

15. 如权利要求14所述的外部存储系统，其特征在于所述彩色矩阵(206)包括多个行和列，该彩色矩阵是按行装载按列读出的。

16. 如权利要求15所述的外部存储系统，其特征在于还包括临时存储包括像素坐标数据在内的像素规格的寄存器装置(202)，所述彩色矩阵(206)是部分地由所述寄存器装置的至少一部分内容寻址的。

17. 如权利要求1所述的外部存储系统，其特征在于还包括记录正被处理的像素是不是正被处理的当前字符的一部分的位未决装置(210)。

18. 如权利要求17所述的外部存储系统，其特征在于还包括随机存取存储器(RAM)(6,8)，和将所述变换电路(52)产生的字符数据根据所述位未决装置(210)的状态传送到所述RAM的装置。

19. 如权利要求1所述的外部存储系统，其特征在于还包括表示无需更新的，与正被处理的像素规格有关字符的各位的装置(210)。

20. 如权利要求19所述的外部存储系统，其特征在于所述变换电路(52)包括响应所述表示装置(210)的规定状态产生字符地址的装置。

21. 如权利要求1所述的外部存储系统，其特征在于还包括存储所述变换电路产生的字符地址的地址寄存器装置(216)，和与地址寄存器装置相连、将所述变换电路所产生的当前字符地址与以前产生的地址进行比较的地址比较器(218)。

22. 如权利要求21所述的外部存储系统，其特征在于还包括存储字符数据的字符存储器装置(6,8)，和响应所述地址比较器以便将存储在所述地址寄存器装置中的地址写出至所述字符存储装置的控制装置(200)。

23. 一种与一显示屏一起使用的图形处理器(2)，其特征在于包

括：

接收像素规格的装置(56,58)；

处理所述像素规格并产生对包括该指定像素在内的字符进行规定的字符规格的变换电路(52)。

24. 如权利要求23所述的图形处理器，其特征在于所述像素规格包括像素的显示坐标和与所述显示坐标相关的彩色信息，而且该处理器还包括接收和临时存储所述彩色信息的彩色寄存器装置(54)。

25. 如权利要求23所述的图形处理器，其特征在于还包括接收从所述彩色寄存器装置(54)来的像素彩色信息的寄存器矩阵(206)。

26. 如权利要求23所述的图形处理器，其特征在于还包括随机存取存储器(RAM)(6,8)，和将字符信息从所述RAM送给所述变换电路以便与由变换电路处理的像素规格关联起来的装置(82，206)。

27. 一种图形处理器(2)，用于具有执行存储在至少一种存储设备(10)中的电视图形程序的主处理单元(20)的信息处理系统(20，19)中，所述图形处理器(2)的特征在于：

接收从所述至少一存储器来的程序指令的装置(INSTR，62，60)；

响应至少一规定的程序指令以便将与所述至少一规定的程序指令相关的基于像素的格式的数据变换成基于字符的数据格式的装置(52)。

28. 如权利要求27所述的图形处理器，其特征在于还包括第一源端公共总线(X)，第二源端公共总线(Y)和目的端公共总线(Z)，变换装置(52)接收从所述第一源端总线和所述第二源端总线来的数据并向所述目的端总线送去数据。

29. 如权利要求27所述的图形处理器，其特征在于所述图形处

理器在使用中与随机存取存储器(RAM)(6,8)相连，还包括控制对所述 RAM 存取的 RAM 控制器(88)，所述变换装置(52)包括：产生字符地址的地址变换装置(202,204,HA,FA,214,216)，产生包括与所述基于像素的数据相对应的数据在内的字符数据的字符数据产生装置(202,206)，和将所述变换装置所产生的字符地址和字符数据传送给所述 RAM 控制器的装置(88)。

说　　明　　书

图形处理器

本发明概括来说涉及一种包括其中配置了可编程处理器的独特的外部存储单元的信息处理装置。本发明具体涉及一种具有程序存储器的可拆卸的外部存储单元，程序存储器中存储的程序一部分由主处理系统如电视游戏系统执行，一部分由为增强主系统的高速图形处理能力而设计的可编程微处理器执行。该可编程微处理器包括将基于像素的格式变换为基于字符的格式的硬件。

本申请与同时由桑等人提交的申请顺序号为____，____题为“用于电视游戏系统等具有可编程图形处理器的外部存储单元”的申请（代理人案卷号为1248—2），和同时由桑等人提交的申请顺序号为____，____题为“用于电视游戏系统等具有增强的存储器控制电路的图形处理器”的申请（代理人案卷号为1248—5）有关。

在电视游戏控制台中配置有8位微处理器和相应的显示处理子系统的已有技术的电视游戏机一般通过在游戏卡中以8位乘8位矩阵形式预先存储字符，并通过建立起利用这些预先存储的字符的各种可编程组合的屏面显示来产生图形。这类已有技术的电视游戏系统一般具有使整个显示背景和游戏者控制的若干“活动目标”或“子画面”移动的能力。

这类已有技术的系统不能出现这样的电视游戏：在这些游戏中包含由多边形组合构成的活动目标，而且这些目标必须能被操纵（如旋转）并在每帧“重新绘制”。这类系统中的已有技术的8位处理器和相关的显示处理电路例如无法进行所需的运算来使三维的、基

于多边形的目标有效地旋转或使这类旋转的目标合适地缩放以产生3—D型特殊效果。本申请发明人认识到精致的图形需要逐个像素地更新屏面和实时地完成复杂的数学运算。这类已有技术的基于字符的电视游戏机是无法胜任这些任务的。

已有技术的8位电视游戏机也不能有效地完成需要逐个像素地迅速更新屏面的其他图形技术。例如，这类系统无法有效地在三维空间使一个目标映像到作为另一个显示目标的一部分所显示的多边形上（此后称为"质地映像"）。

为了超过它有技术的8位机器的图形能力，电视游戏系统已被设计成采用更具功能的16位处理器。这类16位处理器可为电视游戏系统提供处理更精致的图形所需的数学机能。例如，这些系统可生成更高级的彩色和得到更好的图形分辨率。这类16位电视游戏机是基于字符的系统，允许在较大范围上实现可预先绘制成基于字符或子画面的图形的电视游戏。这类16位电视游戏机还允许具有多种彩色的背景平面高速移动并在这些平面的后面或前面配置有活动目标。

然而，这类已有技术的16位电视游戏机在实用上实现不了具有3—D型特殊效果的高级电视游戏，这些游戏要显示由多边形所组成的并需每帧变化的复杂目标。这类已有技术的基于字符的16位机器例如无法在实用中实现需要必须逐帧地放大或缩小的全方位旋转的目标或子画面的游戏。本申请发明人认识到要有效地实现涉及全方位旋转和缩放的基于多边形的目标在内的这类游戏，必需绘制多边形的边界并逐个像素地将合适数据填入这类基于多边形的目标。这类任务必须逐个像素地进行，故要花去大量的时间处理。

在已有技术中，已对可拆卸的游戏卡作了改进，使现行处理器可访问比与主微处理器相联的现有地址线数目所允许的范围更大的程序存储器地址空间，从而得到更高级的游戏。例如，这类已有技

术的8位系统采用了包括执行存储体切换和其他附加功能的多存储器控制器芯片的游戏卡。然而这类有关存储体切换的芯片无法使电视游戏系统胜任上面提到性质的高速图形处理。

本发明针对已有技术的上述问题，提供一种独特的、全编程的、设计成配备在与主信息处理系统连接的可拆卸的外部存储单元中的图形微处理器。在这里所述的示范性实施例中，本发明是在一包括主电视游戏系统和装有图形微处理器的电视游戏卡的电视游戏系统中实现的。

这里所述的图形微处理器和电视游戏系统具有许多独特的、优异的特征，下面综述这些特征。

按照本发明，独特的图形处理器以插件方式与主微处理器连接。为使处理速度最大，图形处理器可以与主微处理器并行地工作。在一示范性实施例中，装有图形协处理器的游戏卡还包括只读存储器(ROM)和随机存取存储器(RAM)。

本发明的图形协处理器对存储事务在它本身需求和从主微处理器读取数据这两者之间作出仲裁。该处理器能与主微处理器同时执行程序以高速处理，在此之前已有技术的电视游戏系统是无法达到的。

本发明的图形协处理器与配置在游戏卡上的三总线结构结合在一起工作，从而通过优化主处理器和卡处理器有效使用这类存储设备的能力以使卡内RAM和ROM存储器得到有效利用。

本发明的全用户编程图形协处理器包括一设计成允许高速处理的独特指令集。该指令集能有效地实行有关3—D图形运算操作，例如拥有由专用硬件执行的用来在主电视游戏系统的字符映像显示中标绘单个像素的专用指令。

该指令集包括独特的基于像素的指令，按程序员的观点，它通过允许对单个像素编址建立一"虚拟的"位映像——尽管主系统是基

于字符的。该像素数据是在扫描时由图形处理器变换为典型地由基于字符的16位主机所采用的格式的字符数据。这样，虽然程序员可用独特的"标绘"指令标绘像素，但有关数据读出到RAM时，数据被变换16位主机所能采用的基于字符的格式。

专用像素标绘硬件执行这指令以有效地实现高速3—D型图形。该标绘硬件有助于将对像素坐标的寻址实时地变换成对主系统所采用形式的字符映像的寻址。

这样，靠指定像素的程序执行图形操作而标绘硬件则在扫描时将各像素的指定值变换成相应格式的字符数据。该字符数据随后映入主处理器图像RAM中供显示用的合适地方。

标绘硬件响应各种有关标绘的指令以允许对某个特定的像素的在显示屏上的X坐标和Y坐标和规定的彩色进行可编程的选择，并标绘相应的像素使X坐标和Y坐标变换成与具有驱动主处理器的图像RAM用的形式的字符定义所对应的地址。

结合附图对下面的本发明示范性实施例的详细说明会使本发明的所有的形式和优点更易于理解。

图1是根据本发明示范性实施例的示范性外部存储系统的方框图；

图2是与目前较佳的示范性实施例的图形协处理器一起使用的示范性主处理系统的方框图；

图3是内置图形协处理器的游戏卡和内置主处理系统的底座部件的示范性物理配置的透视图；

图4A和图4B是根据目前较佳示范性实施例的图形协处理器的方框图；

图5是叙述为起动图形协处理器操作而由主处理系统执行的操作序列的流程图；

图6是图4A所示的算术逻辑单元的更详细方框图；

图 7 是图 4A 所示类型的示范性像素标绘电路的更详细方框图；

图 8A 是示出标绘控制器接收的输入信号和标绘控制器产生的输出信号的方框图；

图 8B 是像素标绘电路的彩色矩阵中的彩色矩阵单元；

图 8C 描述有关像素标绘电路的定时、控制和数据信号；

图 9 是图 4A 所示的 RAM 控制器的更详细方框图；

图 9A 示出有关图 9 所示的 RAM 控制器的示范性定时、控制和数据信号；

图 10 是图 9 所示的仲裁逻辑的电路图；

图 11 是本发明图形协处理器的示范性实施例中的再同步电路图；

图 12 是有关图 11 的再同步电路的定时信号；

图 13 是本发明图形协处理器的 ROM 控制器的更详细的方框图；

图 14 是根据本发明示范性实施例的图形协处理器的超高速缓存控制器的方框图；

图 15A 是本发明图形协处理器中与指令译码有关的电路的方框图；

图 15B 示出说明图 15A 中的先行逻辑的工作用的示范性定时信号；

图 16 和 17 是示出根据本发明示范性实施例的图形协处理器的寄存器控制逻辑的方框图；

图 18 是叙述在完成多边形生成任务中图形协处理器的操作程序的流程图；

图 19、20 和 21 是用来说明按照本发明示范性实施例可以生成的基于多边形的目标的缩放和旋转特征的示范性显示。

根据本发明的示范性实施例，本发明的图形协处理器与任天堂(美国)股份有限公司在市场销售的商品名为超级任天堂娱乐系统(Super NES)的16位电视游戏系统交互作用。该超级任天堂娱乐系统在1991年4月10日提交的申请号为07/651,265的题为"视频处理装置"的专利申请和1991年8月26日提交的申请号为07/749,530的题为"存储器直接存取装置和其所使用的外部存储设备"中得到部分的说明。这些申请通过引用而明显地归并在本申请中。应该理解本发明不仅仅限于与Super NES有关的应用，还可以用于别的电视游戏系统或别的非电视游戏用的信息处理系统。

仅为了引用的方便，根据本示范性实施例的图形处理器在此之后称为"Mario芯片(Mario Chip)"。目前较佳的示范性实施例中所述的Mario芯片封装在电视游戏卡中。应该理解只要Mario芯片在使用时与程序存储器和主处理单元连接，将Mario芯片与程序存储器置于相同卡盒内并不是本发明所必须的。

图1示出根据本发明示范性实施例的示范性电视游戏卡/外部存储系统。该游戏卡具有其上安装有图1全部元件的印刷电路板(图上未画出)。该卡具有排列在印刷电路板插入端的接插件电极阵列1以将信号传送到或传送出Super NES主控制台。接插件电极阵列1由安装在Super NES主控制台内的相配的接插件容纳。

根据本示范性实施例，配备在游戏卡上的Mario芯片(图形协处理器)2是100至128插脚的集成电路芯片。Mario芯片接收许多从主处理系统(例如Super NES)来的控制、地址和数据信号。例如，Mario芯片由插脚P112接收来自主处理系统的21 MHz时钟输入，经插脚P117接收可以为21 MHz(或另一预定频率)的系统时钟输入。该系统时钟输入例如可用来向Mario芯片提供用于主CPU存储器存取的存储器定时信息和提供为Mario芯片内的操作定时的时钟信息。Mario芯片2还包括可选配的外部时钟输入(插脚

P110)，将 Mario 芯片与外部晶振 4 连接起来，以比从主系统接收的 21 MHz 更高的时钟频率驱动 MarioCPU。

主 CPU 的地址输入(HA)从主处理系统（例如 Super NES/图像处理单元 PPU)的地址总线经插脚 P37 至插脚 P62 连接到 Mario 芯片 2。来自主系统的数据输入(HD)类似地从主 CPU 数据总线经插脚 P65—P72 连接到 Mario 芯片 2。另外 Mario 芯片 2 经 P119 从主 CPU 接收存储器更新信号 RFSH，经插脚 P118 接收复位信号，经插脚 P104、P105 接收读出和写入控制信号。该 Mario 芯片可产生中断请求信号 IRQ 并将该信号 IRQ 经插脚 P120 连接到 Super NES。也可从 Super NES 接受其它控制信号，例如经插脚 P106 接收可起动对主程序 ROM 10 访问的 ROM SEL 信号。另外该游戏卡拥有与 Super NES 的认证处理器在输入线 I、输出线 O 和复位线 R 交换数据的认证处理器 3 。用来认证游戏卡的认证处理器 3 和安全系统可以为美国专利 USP 4,799,635 示出的那种，该专利通过引用而归并于此。

Mario 芯片经 RAM 地址总线(RAM A)、及 RAM 地址插脚 P74—P91 和 RAM 数据总线(RAM D)及数据插脚 P93—P100 与 RAM 6 和 RAM 8 连接。这些 RAM 可以是利用分别经插脚 P90—P91 连接的行地址和列地址选通信号(RAS，CAS)进行部分控制的动态存储器件。可采用一个或多个静态 RAM 来取代动态 RAM，这时插脚 P90 和 P91 将被用来在没有行地址和列地址选通信号的情况下，将地址信号连接到它们各自的 RAM。允许写入控制信号 WE 可经插脚 P107 相应地送到 RAM6 和 RAM8。

读、写控制信号(R，W)由主 CPU 产生并经插脚 P104 和 P105 与 Mario 芯片连接。Mario 芯片通过监测这些读出和写入线能够确定 Super NES CPU 所要执行的存储器存取操作的种类。实质上来自主机的所有地址线和控制线都类似地由 Mario 芯片监测，以掌握

主CPU所要进行的操作。由Mario芯片接收的ROM和RAM寻址信号受到监测，并将它们送到相应的存储装置。在这方面，ROM地址经ROM地址总线和插脚P2至P26与程序ROM10相连，而RAM地址经插脚P74至P91与RAM 6和RAM8相连。从主CPU出来的ROM和RAM数据输入分别经ROM数据总线和插脚P28—P35送到ROM10和经RAM数据总线和插脚P93至P100与送到RAM6和RAM8。

应该认识到Mario芯片除了与这里所述的ROM和RAM外还可以与较宽度范围的不同存储设备结合一起使用。例如可以期待Mario芯片与采用CD ROM的电视游戏系统便利地结合一起使用。

例如在图1中不是采用ROM 10，而是使用CD ROM（未图示）来存储字符数据，程序指令，图像、图形和声音数据。将一种常规类型的CD机座（也未图示）与Mario芯片2合适地连接起来，通过地址总线P2—P26为存取数据接收存储器地址信号和/或通过数据总线P28—P35接收指令。至于CD机座和CD ROM存储系统具体的结构和操作细节是本技术领域的技术人员众所周知的。CD ROM存储带来的优点在于使每字节信息的存储成本显著地降低。存储数据的成本可以比存储在半导体ROM上低百分之一百到一千。不幸的是CD ROM存储器存取/读出时间比半导体ROM的还长。

Mario芯片采用在至少三条总线上的信息允许并行利用的三总线结构。在这方面，在图1所示的游戏卡中，Mario芯片同ROM总线（包括ROM数据线、ROM地址线和控制线），RAM总线（包括RAM地址线、数据线和控制线）和主处理器总线（包括主机地址线、数据线和控制线）相连。

Mario芯片结构允许发生流水线操作以优化吞吐量。在这方面，Mario芯片可以在从ROM读取一个数据字节时，同时处理其它数

据，还同时向RAM写入另外的数据，因而能非常有效地处理有关3—D型的图形。如下面进一步说明的那样，Mario芯片2内部采用16位结构，还设计为与8位的ROM 10和RAM6、RAM8等芯片接口。在内部，所有的内部数据总线和内部寄存器均为16位。从ROM 10读出和写入RAM 6、RAM8均被"缓冲"并且一般不会放慢程序执行。

类似地，Mario芯片2可以从CD ROM存取指令和图形数据并将该信息写入RAM6、RAM8以便以后通过DMA（存储器直接存取方法）转送到主处理器（例如Super NES的图像处理单元PPU）的图像RAM中。本技术领域的技术人员会理解可以对Mario芯片2进行编程绕过RAM的存储和存取操作而协调从CD ROM直接转送数据给PPU的图像RAM。

尽管CD ROM的读出存取时间较长，但Mario芯片2极快的处理速度使CD ROM存储可实际用于图形应用。视频和音频数据在存储于CD ROM上之前采用常规的数据压缩技术进行压缩。数据压扩技术对本领域技术人员是众所周知的。在从CD ROM取出压缩的数据之后，Mario芯片2运用常规的数据解除压缩算法使数据扩展所需时间比常规图形处理器所能达到的短得多。Mario芯片2用21 MHz时钟信号进行工作，故而Mario芯片2可在规定期间内完成扩展以转送给RAM6、RAM8。

这样有大量的视频和音频数据（以压缩形式）在典型的CD ROM存取周期内存取。由于在Mario芯片2进行数据扩展后，每数据字节的实际存取时间显著地减小，因而使其相对长的存取时间的影响减至最小。在Mario芯片2执行扩展的同时，主图形处理例如Super NES PPU可自由地进行其他处理任务。当然，对于某一特殊应用，如果速度不成为其问题，Mario芯片2可以以不压缩的形式从CD ROM存取数据。

当使用静态 RAM 时该游戏卡还可以包括一后备电池。后备电池 12 经电阻 R 连接到后备电池电路 14 以在电源失效时提供静态 RAM 的后备电压(V_{SRAM})和静态 RAM 芯片选择信号 RAMCS 以保全数据。

另外，在选配项设定电阻 16 与 RAM 地址总线连接。在平常操作中，Mario 芯片地址线输出到 RAM6 和 RAM8。然而在复位或加电操作期间，这些地址线根据它们是连在预定的电压 V_{cc} 还是接地而用作输入线以产生高或者低信号。在这种方式下，"1"或"0"合适地读进内部的 Mario 芯片寄存器。在复位后，Mario 芯片(在执行程序期间)根据这些电阻的置值，例如可确定与 Mario 芯片关联的乘法器时钟频率、RAM 的存取时间、要用于 Mario 芯片内的其他操作的时钟频率等。Mario 芯片通过利用这些选配项设定电阻，例如，无需对 Mario 芯片的设计作任何改动，就可适合同若干不同类型的存储设备一起使用。例如，如果检测出动态 RAM 置值就要在各相应时刻加上更新信号。另外，可选配设定可以用来控制例如处理器乘法器电路的运行速度，和允许图形处理器以比所能执行某些乘法指令还快的速率执行到的指令。这样，通过启动延迟的乘法操作，能以比其它方法可用的频率更快的时钟频率运行余下的指令(例如处理器时钟为 30 MHz，而可选配设定会有效地以 15 MHz 来执行乘法指令)。

图 2 是设计为与图 1 中示出的示范性游戏卡相连的示范性主电视游戏系统的方框图。图 2 可以代表例如由美国任天堂目前销售的 Super NES。但本发明不限于与 Super NES 有关的应用或具有图 2 所示的这种方框图的系统。

该 Super NES 在控制台 20 内包括例如可以是 65816 兼容型微处理器的 16 位主 CPU。CPU22 与例如可以包括存储容量 128 K 字节的工作 RAM 32 相连。CPU22 与图像处理单元(PPU)24 相连，

PPU 又与例如可包括存储容量 32K 字的图像 RAM30 相连。CPU22 在垂直或水平回扫消隐间隔期间经 PPU24 对图像 RAM 进行存取。这样，CPU 22 只能在 PPU24 对图像 RAM 存取在进行线扫描期间以外的时间通过 PPU24 对图像 RAM30 进行存取。PPU24 从图像 RAM30 产生在用户电视 36 上的图像显示。CPU 还与连接到工作 RAM28 上的音频处理单元相连。可以包括市售的音响芯片在内的 APU26 产生与存储在游戏卡上 ROM10 内的电视游戏程序相关的声音。CPU22 只能经 APU26 对工作 RAM28 进行存取。PPU24 和 APU26 经射频调制器单元 34 与用户家用电视 36 连接。

Super NES 中的图像 RAM30 必须装载存储在游戏卡中的程序 ROM10(它不仅存储游戏程序，还存储玩游戏时所用到的字符数据)内的合适的字符数据。任何活动目标，譬如所要显示的子画面信息、或背景信息在使用前必须存入图像 RAM30 内。该程序 ROM10 经与图 1 中示出的印刷电路板边缘接插件相连的相配的接插件 18 由 CPU22 存取。PPU 经共用的主 CPU 数据总线、地址总线和接头 23 与游戏卡连接以提供让 PPU 数据和控制信号与游戏卡相连的通路。APU26 经共用的主 CPU 总线和音频总线 27 与游戏卡连接。

CPU22 地址空间的分配，是使得程序 ROM 10 存储单元从位置 0 开始并一般分成 32K 字节的各个区段。程序 ROM 使用近一半的 CPU 地址空间。每个 CPU 地址空间 32K 字节区段的顶端位置一般用来对工作 RAM32 和各种寄存器编址。程序 ROM10 一般是 4 兆字节。用在 Super NES 中的 CPU22 能够对程序 ROM10 的全体进行寻址。另一方面，Mario 芯片 2 只包括一 16 位程序计数器，因而还包括用以选择程序 ROM10 中的 32K 字节存储区的存储区寄存器。

在本示范性实施例中，Mario 芯片具有与 Super NES 存储器分配图相对应的全 24 位地址空间。在从位置 $00：8000 开始的位置

上含有ROM10，而游戏卡上的RAM芯片6、8则从位置$70：0000开始。

既然游戏卡上的ROM10和RAM6，RAM8连在分开的总线上，他们就能由Mario芯片并行存取。而且对RAM6，RAM8的存取频率比对ROM快，Mario芯片就是设计成利用这性能优点的。Mario芯片无法对Super NES中的任何存储器进行存取，即无法对工作RAM 32或PPU图像RAM 30存取。

为使Mario芯片处理数据或输入位映像中，数据必须存放在该Mario芯片的RAM芯片6、8内。这样，在NES CPU程序和马里奥芯片程序间共用的任何变量一定要在Mario芯片的RAM芯片6、8内。Mario芯片需要用的任何预先存储的数据可以在ROM10中，任何变量则在RAM6、RAM8中。

仅仅是Super NES所需要的任何专用变量无需放在游戏卡RAM6、RAM8中。事实上，RAM6、8的存储空间很宝贵，所以最好将最优先需要的内容分配给RAM6、RAM8。任何非必需的变量应该存储在Super NES的内部RAM32中。

Mario芯片写入的位映像是Mario游戏卡RAM6、RAM8中，并当每一帧位映像已被全部提供时，在Super NES的控制下，通过DMA转送到PPU的图像RAM30中。

Super NES的CPU22对Super NES控制台内的所有内部RAM进行存取，就象没有Mario芯片。Mario芯片无法对这RAM进行存取，故而必须由CPU本身引入MarioROM/RAM芯片和内部Super NES RAM间转送的所有数据。通过CPU 22编程就可传送数据，或通过DMA传送来移动数据块。对于所有的游戏程序，Mario卡ROM10和RAM6、RAM8都以惯用的方式映入。

CPU 22控制哪一个CPU可临时对游戏卡ROM和RAM芯片进行存取。在加电或复位的情况下，Mario芯片被关断，完全由

CPU22 对游戏卡 ROM 和 RAM 芯片进行存取。为使 Mario 芯片运行程序，有必要使 CPU22 程序放弃对 ROM 或 RAM 芯片，最好是对两者的存取，而等待 Mario 芯片完成其所交给的任务，或者 CPU22 可将某一程序复制进内部的工作 RAM32 并在那儿执行它。

Mario 芯片有若干可从 Super NES CPU 侧编程和读出的寄存器。这些均被映入从位置 $00：3000 开始的 CPU22 存储器分配图。

如图 2 所示，Super NES 产生和接收种种控制信号。当 Super NES CPU22 需对程序 ROM10 进行存取时，它产生一控制信号 ROM SEL。要开始存储器更新，该 Super NES 产生一更新信号 RFSH。当 Mario 结束某一操作时，它发送一中断信号到与 Super NES CPU 相联的中断请求线上。CPU22 另外还产生读出和写入信号。

系统定时信号是从控制台 20 中的定时连锁电路 21 产生出的。加电/复位信号也是在主控制台 20 中产生并送到游戏卡。

Super NES 还包括认证处理设备 25，它按照上面提到的美国专利 USP 4,799,635 同游戏卡上的认证处理设备 3 在输入线 I、输出线 O 和复位线 R 上交换数据。美国专利 USP 4,799,635 指出，该处理设备 25 使 CPU22 处于复位状态直到认证被证实。

在图 2 中用方框形式表示的 Super NES 电视游戏机在此只是总体上的描述。有关包括 PPU24 在内的 Super NES 的进一步细节，例如可以在 1991 年 4 月 10 日申请的申请顺序号为 07/651,265 的题为“视频处理装置”的美国专利申请中找到，该申请通过引用而明显地归并在本申请中。诸如 Super NES 和游戏卡之间的信息如何传送之类的再进一步细节可以在 1991 年 8 月 26 日申请的申请顺序号为 07/749,530 的题为“图像处理系统中的存储器直接存取装置和其所使用的外部存储设备”的美国专利申请和 1991 年 11 月 19 日申请的申请顺序号为 07/793,735 的题为“镶嵌图像显示装置及所用的外

部存储单元"的美国专利申请中找到，这些申请在此引用而归并在本申请中。

本申请发明人认识到在一些应用中，需要利用这种主处理器DMA 电路在垂直回扫消隐期间传送的信息比实际可能传送的多。因而即使会导致画面尺寸略有收缩，还是希望延长垂直回扫消隐时间。通过采用这一方法，在处理速度和画面更新频率方面就体现出显著的优点。

图 3 示出容纳图 1 所示的 Mario 芯片和游戏卡上其它结构的游戏卡盒的示范性机械外观设计的透视图。图 3 类似地示出容纳图 2 示出的 Super NES 电视游戏硬件的电视游戏控制台 20 的示范性外部机座的透视图。这类电视游戏控制台 20 和相关的可卸下的游戏卡 19 示于 1991 年 8 月 23 日申请的申请顺序号为 07/748,938 题为"TV 游戏机"的美国专利申请的图 2—9 中，该申请在此引用而归并在本申请中。

图 4A 和 4B 是示于图 1 中的 Mario 芯片 2 的方框图。首先集中看示于图 4A 和 4B 中的各种总线，指令总线 INSTR 是将指令代码送到各种 Mario 芯片元件的 8 位总线。X、Y 和 Z 总线是 16 位数据总线。HA 总线是 24 位的主系统地址总线，在目前较佳实施例中，使用时与 Super NES 的地址总线相连。HD 总线是使用时与 Super NES 的数据总线相连的 8 位的主系统数据总线。PC 总线是将 Mario 芯片程序计数器(即通用寄存器块 76 中的寄存器 R15)的输出送给各种系统元件的 16 位总线。ROM A 总线是 20 位 ROM 地址总线。ROM D 总线是 8 位 ROM 数据总线。RAM 总线是 16 位 RAM 地址总线。RAMD IN 总线是 8 位 RAM 读出数据总线，RAMD OUT 总线是 8 位写入数据总线。

Mario 芯片和 Super NES 共用游戏卡 RAM6、RAM8，起到在 Mario 芯片和 Super NES 之间传递数据的主要机能。Super NES 给

地址总线 HA 和数据总线 HD 对 Mario 芯片进行存取。Mario 芯片的寄存器 76 由 Super NES 经 Super NES 的地址总线 HA 进行存取。

Super NES 经 Mario 芯片 2 对游戏卡上程序 ROM10 和 RAM6、RAM8 存取。ROM 控制器 104 和 RAM 控制器 88 接收到由 Super NES 产生的有关存储器存取的信号，就分别起到 ROM 和 RAM 的存储器存取。顺便解释一下，RAM 选择信号 RAMCS 被 Mario 芯片用来确认 Super NES 正要对该 RAM 寻址。

示于图 4A 和 4B 中的 X、Y 和 Z 总线是内部的 Mario 芯片数据总线。X 和 Y 总线是源数据总线而 Z 数据总线则是目的总线。这些总线载送 16 位并行数据。

当执行指令时，Mario 芯片 2 可以将指令用的源数据放在 X 和/或 Y 总线上面结果数据放在 Z 总线上。例如，在执行将两寄存器的内容相加并将结果放在第三寄存器的指令时，算术逻辑单元(ALU)50 经 X 和 Y 总线接收到两个源寄存器的内容并将结果送到 Z 总线(转而送到块 76 中的指定寄存器)。由 Mario 芯片 2 的指令译码电路 60 对指令操作码译码得出的控制信号被送到 ALU50 起动加法(ADD)操作。

正如针对图 1 说明所注解的那样，与 Mario 芯片相连接的 ROM 总线、RAM 总线和 Super NES 主机总线能并行地进行信号通讯。Mario 芯片 2 监视经过主机 Super NES 总线转输的控制、地址和数据信号以确定主系统正在执行的操作。根据任何给定的时刻正在执行的 Super NES 操作可以并行地对游戏卡 ROM 总线和游戏卡 RAM 总线进行存取。在通常的 Super NES 游戏卡中，主机 CPU 的地址线和数据线直接与 ROM 和 RAM 相连，这样 RAM 和 ROM 就不能被并行地存取。

根据本发明的一个方面，Mario 芯片 2 使图 1 中所示的 ROM

总线和 RAM 总线与 Super NES 总线在物理上分开。Mario 芯片 2 监视在 Super NES 总线上传输的信号，并确定什么信号需经不是分时的两条分开的 ROM 总线和 RAM 总线送到 ROM 芯片和 RAM 芯片。通过分开 ROM 总线 RAM 总线，Mario 芯片能同时从 ROM 读出和向 RAM 写入。在这种方式下，Mario 芯片可以用便宜的其存取时间比 RAM 存取时间慢得多的 ROM 芯片有效地工作而无需等到 ROM 存取结束后才对 RAM 存取。

回到图 4A，Mario 芯片如前面所注明的那样，是全编程处理器并包括 ALU 50。除了乘法操作由乘法器 64 执行和某些像素标绘操作由标绘硬件 52 执行外，ALU50 执行 Mario 芯片中所配备的所有算术功能。一收到出自指令译码器 60 的合适控制信号，ALU50 就执行加法、减法、异或、移位和其他操作。如图 4A 所示，ALU 50 从 X、Y 总线上接收所要操作的信号，执行从指令译码器 60 收到的控制信号所起动的操作，并将该操作的结果送到 Z 总线。下面将结合附图 6 进一步详细地说明 ALU。

Mario 芯片 2 另外包括专用硬件以便有效执行 3 —D 型特殊效果和其他图形操作，使运用这些性能的电视游戏可以实现。在这方面，Mario 芯片 2 包括标绘硬件 52，它辅助将对像素坐标的编址实时地变换成 Super NES 所采用的那种对字符映像的编址。好处是可以通过指定定义各像素在显示屏上的位置的 X 和 Y 坐标来对 Mario 芯片进行编程。

这样，图形操作是根据程序员指定像素来执行的，标绘硬件电路 52 在扫描时将各像素指定值变换成正确格式化的字符数据。该字符数据随后映入示于图 2 中的 Super NES 图像 RAM30 中供显示用的符合要求的地方。在这种方式下，Mario 芯片程序员只需将 Super NES 的图像 RAM 30 看作位映像，而实际上它是字符映像。

该标绘硬件 52 响应种种有关标绘的指令，允许对一特定像素

在显示屏上的X和Y坐标和预定彩色进行可编程的选择，和标绘相对应的像素使X和Y坐标变换成与具有用来驱动Super NES图像RAM 30的形式的字符定义相对应的地址。

标绘硬件52具有相关的数据锁存器，允许在写到游戏卡RAM前缓冲尽可能多的像素数据，以使RAM数据事务尽量减少。在X和Y坐标数据在标绘硬件52中变换和缓冲之后，字符定义数据随后传送到游戏卡RAM。

该标绘硬件52分别经PLOT X寄存器56和PLOT Y寄存器58接收X、Y坐标数据。在目前较佳实施例中，PLOT X和PLOT Y寄存器不是分开的寄存器（如图4A所示）而是Mario芯片通用寄存器(即示于图4B中的寄存器块76中的寄存器R1和R2)。

该标绘硬件52还经彩色寄存器54接收像素彩色信息。如下面将会进一步说明的那样，所显示的各像素的彩色存储在8×8寄存器矩阵中，每个像素的彩色说明占据该矩阵的一列。

标绘硬件52处理字符号地址和与X、Y和色彩输入有关的数据并将它们送到字符RAM 6、RAM 8。字符地址经输出线53送到RAM控制器88和RAM地址总线RAM A。字符数据经输出线55、多路转换器93和RAM数据总线RAM D DUT送到字符RAM。该标绘硬件52在与Super NES字符格式保持兼容的同时允许字符中的像素单独地编址，以藉此提供给程序员一"虚拟的"位映像显示系统。"虚拟的"位映像保存在游戏卡RAM中并在每帧显示结束时利用例如上面提到的申请顺序号为07/749,530的美国专利申请中的DMA电路传送到Super NES的图像RAM30。该标绘硬件52能高速控制单个像素，使涉及到旋转和缩放目标的某些3—D型图形效果得以实现。

由于要把像素格式变换成字符格式，标绘硬件52还经RAMD IN和数据锁存器82和输入线83从卡RAM 6、RAM 8接收与当前

像素X、Y附近的其他像素有关的信息。通过使用从RAM 6、RAM 8检索出的和暂时存储在RAM数据锁存器中的以前的像素数据，可以使对RAM的写入次数降为最小。示于图4A中的RAM数据锁存器80、84和86还用来缓冲收到的关于某一像素的彩色数据(它们已存储于游戏卡RAM内的多重位平面中)，以将这些数据提供给标绘硬件52。

RAM数据锁存器80与Super NES数据总线相连因而Super NES能读出该数据锁存器的内容。RAM数据锁存器80、82和86由RAM控制器88控制。RAM数据锁存器84和86用来接收来自RAM 6、RAM 8的数据并将来自RAM 6、RAM 8的数据送到目的端总线Z以便于装载进寄存器块76中的预定寄存器。另外与RAM控制器88相连的锁存器90是缓冲RAM地址的。RAM控制器88利用存储在锁存器90中的地址经RAM A总线对RAM 6、RAM 8寻址。RAM控制器88还可以由Super NES经地址总线HA进行存取。

标绘硬件52还响应读出像素(READ PIXEL)指令，它读取寄存器R1的内容所定义的水平位置和寄存器R2的内容所定义的垂直位置上的像素的彩色信息，并经目的端总线Z和输出线87将结果存储在寄存器块76中的预定寄存器中。将结合图7、8A和8B和说明对PLOT硬件52作进一步说明。

流水线缓冲寄存器62和ALU控制指令译码器60与指令总线INSTR相连以产生控制信号(应用于整个Mario芯片)来起动与置于指令总线的命令相应的操作。Mario芯片2是在其执行当前指令的同时就读取所要执行的下一个指令的流水线微处理器。流水线寄存器62存储所要执行的下一个指令，使得如果可能的话就在一个周期中执行各指令。置于指令总线上的指令是按存储于寄存器(例如可以是示于图4B的寄存器块76中的寄存器R15)中的程序计数器

的内容寻址的。

由 Mario 芯片 2 执行的指令可以从示于图 1 中的程序 ROM 10 或 Mario 芯片的内部超高速缓存器 RAM 94 得到，或从游戏卡 RAM6、RAM 8 得到。如果正从 ROM 10 执行程序，ROM 控制器 104(示于图 4B 中)将读取指令并将它置于 Mario 芯片指令总线 INSTR。如果程序指令是存储在超高速缓存 RAM 94 中，则指令会经超高速缓存 RAM 输出总线 95 直接从超高速缓存输出置于指令总线上。

对主 CPU (即 Super NES)的编程，是使程序 ROM 10 部分分配给 Mario 芯片程序指令。Super NES 程序命令 Mario 芯片执行一规定的功能，然后将存取 Mario 芯片程序代码的 ROM 10 中的地址提供给 Mario 芯片。流水线寄存器 62 读取正执行着的指令的下一字节指令，以将有关指令的信息提供给指令译码器 60，使译码器能在程序执行当中先行了解即将发生的事件来预期有关处理。部件 60 中的译码和控制电路产生命令 ALU 50、标绘硬件 52、超高速缓存控制装置 68 等的控制信号，以完成由正执行着的指令代码所指示的操作。

Mario 芯片还包括一个同 ALU50 分开的高速、并行乘法器 64。乘法器 64 响应规定的指令操作使从 X 和 Y 源端总线接收到的两个 8 位数相乘并将 16 位结果装上目的端总线 Z。如果可能该乘法操作在一个周期内完成。输入到乘法器 64 的两个数可以带符号或不带符号。乘法器还能完成长字乘法操作，使两个 16 位数相乘而产生一个 32 位的结果。乘法器还包括相关联的部分乘积寄存器 66 来存储乘法操作期间产生的部分乘积。当译出一个乘法操作代码时，就由指令译码器 60 输出的控制信号启动乘法器 64。乘法器 64 将在最小 4 个时钟周期内执行涉及 16 位字的乘法的长字乘法指令。

长字乘法指令具有以下格式：

R_4(低位字)，DREG（高位字）=Sreg * R6

执行该指令就使源寄存器乘以寄存器R6的内容并将32位结果存储在寄存器R4/DREG（低/高位）。乘法是带符号的并对32位结果设定零和符号标志。

该操作按照以下六个步骤进行：

步骤1：不带符号的乘法 R4[0…15]=SREG[0…7] * R6[0…7]

步骤2：X带有符号。R4[0…15]=R4[0…15]+256 * SREG[8…15] * R6[0…7]。不管乘积的高8位，但保留加法的进位。

步骤3：X带有符号。R5[0…15]=CY+(R6[8…15] * SREG[0…7])÷256；加上符号。

步骤4：X不带符号，Y带有符号。

R4[0…15]=R4[0…15]+256 * SREG[0…7] * R6[8…15]。不管乘积的高8位，但保留加法的进位。

步骤5：Y带有符号，R5[0…15]=R5[0…15]+CY+SREG[0…7]+R6[8…15])÷256；加上符号。

步骤6：X、Y带有符号，R5[0…15]=R5[0…15]+RY[8…15] * R6[8…15]。

用在本示范性实施例中的乘法器64例如可以是麦格劳-希尔1984年出版的由卡瓦诺著的《数字计算机算术》中所描述的那种。

参见图4B，超高速缓存控制器68(在图14中会进一步详细图示)允许程序员有效地起动装载操作，将想要高速执行的那部分程序装入超高速缓存RAM 94。这种“超高速缓存(CACHE)”通常应用于在图形处理中频繁发生的小程序循环。Mario芯片指令集包括“CACHE”指令。紧接着该CACHE指令的任何指令被装载进超高速缓存RAM，直到超高速缓存RAM满载为止。当执行CACHE指

令时，当前程序计数器状态被装载进超高速缓存基址寄存器 70。这样，该超高速缓存基址寄存器的内容规定了超高速缓存开始的起始位置。

大多数指令在一个周期内执行。来自相对慢的外部存储器象 ROM 10 和或 RAM 6、8 的指令必须在执行它们之前就取出。这将花费另外大约 6 个周期。为提高程序执行速度，应该利用在 Mario 芯片内的"超高速缓存"RAM 本身。

超高速缓存 RAM 可以是一 512 字节的指令超高速缓存器。这容量与平均程序的大小相比是比较小的；因而程序员必须对如何最大限定地利用超高速缓存 RAM 作出决定。能够装入 512 字节超高速缓存容量的任何程序循环均能以全速运行，使读取和执行均在一个周期内进行。因为总线是分开的，故而在执行来自内部的超高速缓存 94 的代码时能同时对 ROM 和 RAM 存取。

超高速缓存 RAM 92 可以通过在它执行旋转和缩放计算时，在它利用 PLOT 指令(将在下面说明)将像素写入 RAM 6、RAM 8 时，同时运行超高速缓存中会从 ROM 10 读出各像素色彩的循环，来方便地使子画面旋转。所有这一切都是并行地发生的，尽管最慢的操作降低了速度但仍给出很快的吞吐率。最慢的操作通常是 ROM 数据的读取，这就是 Mario 芯片为什么被设计成利用对 ROM 和 RAM 缓冲存取的原因。

与运行来自相对慢的 ROM 的程序相比，运行来自超高速缓存 RAM 94 的程序速度可快约 6 倍，但它首先必须从 ROM 装载到超高速缓存中。这是通过将指令置于所要超高速缓存的任何循环的起始部分来进行的。从 CACHE 指令地址起，将只超高速缓存循环的前 512 字节。在执行循环第一次迭代的代码时，程序就从 ROM 10 出来并以 60 字节块形式复制进超高速缓存 RAM 。该循环所有进一步迭代都将来自超高速缓存 RAM 而不是 ROM 10。

CACHE 指令能随意地用在任何重复的程序循环前面。只是循环的后续迭代能得益于其程序置在超高速缓存中。如果程序循环大于 512 字节并溢出超高速缓存 94，它仍将正确地工作，只不过前 512 字节从超高速缓存 94 运行而剩下的照常将从 ROM 10 运行。这部分地提高了速度但不够理想。

在较佳实施例中作为超高速缓存控制器的一部分的超高速缓存标志位寄存器 72 对超高速缓存 RAM 94 中已装载的存储器位置进行标识。该超高速缓存标志位允许 Mario 芯片迅速地判定某一程序指令是不是能从更快的超高速缓存 RAM 执行而不是从程序 ROM 10 执行。可以由超高速缓存控制器 68 或 Super NES 经 Super NES 地址总线 HA 经多路转换器 96 对超高速缓存 RAM 94 进行存取。

超高速缓存控制器 68 与程序计数器总线 PC 相连以装载超高速缓存基址寄存器 70 并执行超高速缓冲存储器地址超出范围的校验操作。

与从 ROM 10 读出可达到的并行度相似，Mario 芯片还提供并行地写入到 RAM 6、RAM 8 的途径。不管什么时候只要 Mario 寄存器被写入到 RAM 6、RAM 8，它就起动例如在 RAM 控制器 88 中的单独的 RAM 写入电路进行存储事务。这一般将花去 6 个周期，但假使程序员避免在这样做时进行另一个 RAM 的事务，就不会使该处理器延迟。例如在两个存储指令之间插入别的处理就能进行得更迅速。那样 RAM 写入电路就有时间进行它的工作。如果连续用到两个写入，那么在进行第一个写入时第二个会使处理器延迟。

例如(使用下面将要说明的指令集中的指令)：

```
FROM    R8          ;将 R8 存入(R13)
SM      (R13)
SM      (R14)       ;将 R0 存入(R14)
TO      R1
```

```
FROM    R2
ADD     R3          ;执行:r1=r2+r3
TO      R4
FROM    R5
ADD     R6          ;执行:r4=r5+r6
```

注意这两个存储指令互相太接近了。由于RAM总线正忙于试图完成第一存储指令因而第二个将多花去6个周期。

运行速度会更快的写入代码的较佳途径是由其他有用的代码间隔开这两个存储指令。例如：

```
FROM    R8          ; 将R8存入(R13)
SM      (R13)
TO      R1
FROM    R2
ADD     R3          ;执行:r1=r2+r3
TO      R4
FROM    R5
ADD     R6          ;执行:r4=r5+r6
SM      (R14)       ;将R0存入(R14)
```

在这种方式下，在第一存储指令导致向RAM写入的同时可并行执行若干个指令，再在几个周期之后进行第二个存储操作。

下面说明的指令集包括一将寄存器写回到最后所用的RAM地址的快速指令。这通过从RAM取出值，对它进行一些处理，随后将它快速存回，故而允许"成批的"数据处理。

参见图4B，一直接数据锁存器74与指令总线相连。这一数据锁存器允许指令本身提供数据源，这样就不需要由指令指定源端寄存器。该直接数据锁存器74的输出与目的端总线Z相连。它转而与寄存器块76的一个规定的寄存器相连。指令译码电路60对"直接"

数据指令译码并开始执行相应的向寄存器传送的操作。

示于图 4B 中的 GET B(取字节)寄存器 98 是与前面所述的延迟/缓冲读出操作一起使用的。在这方面，已有技术的处理器由于广泛地利用存取时间相对慢的 ROM，只要执行到 ROM，一般就不得不等待它完成数据读取。而利用下面说明的延迟/缓冲读取机制，在完成数据读取时还可执行其他操作。按照该机制，不管以什么方法对寄存器块 76 中的寄存器 R14 存取或修改，就自动地在 R14 内容所标识的地址起动对 ROM 或 RAM 的读取。

如图 4B 所示，寄存器 R14 与 ROM 控制器 104 相连。不论何时，以任何方式修改寄存器 R14 的内容，ROM 控制器 104 就起动对 ROM 的存取。对 ROM 存取的结果经与 ROM 数据总线 ROMD 相连的多路转换器装载到 GET B 寄存器 98 中。下面列出的指令允许对在 GET B 寄存器 98 中缓冲的信息进行存取。该信息经多路转换器 100 装上目的端总线 Z 随后进入到寄存器块 76 中的某个寄存器。

在这种方式下，若已知从 ROM 读取数据要花去一定个数的处理周期，就可起动那读取操作并在起动了这数据读取之后，Mario 芯片不是等着不执行其它操作，而是能执行例如与数据读取无关的代码。GET B 寄存器 98 还可以用来存储经如图 4B 所示的多路转换器 102 从 RAM 6、RAM 8 检取出的信息。

在寄存器块 76 中配备有十六个 16 位寄存器(R0—R15)。寄存器 R0—R13 是通用寄存器(尽管其中某些寄存器往往用于下面所说明的专用目的)。如上所述，寄存器 R14 被用作为读出存储器的指针，当被修改时，就起动从 ROM (或 RAM)的读出周期。读出的字节被存储在临时的缓冲器(GET B 寄存器 98)以便以后由 GET L (取低字节)或 GET H(取高字节)命令存取。寄存器 R15 是程序计数器。在每个指令开始执行时它指向正要存取的下一个指令。

寄存器R0是一通用寄存器，一般用作累加器。对于大多数单个周期的指令它还是缺省的源端和目的端寄存器。例如，若想要使R0和R4的内容相加在一起就只须明显地指定寄存器R4。

当执行循环指令时就专用寄存器R11、R12和R13。寄存器R13存储循环顶端所要执行的指令的地址，而寄存器R12存储循环所要执行的次数。若寄存器R12的内容非零，则将在R13内容所指定的地址的指令装到程序计数器(R15)并且执行。寄存器R11存储循环完成后所要返回的地址。

寄存器控制逻辑78与寄存器块76相连并控制对通用寄存器R0—R15的存取。根据正要执行的特定指令的格式，指令译码逻辑60将指定一个或多个寄存器R0—R15。寄存器控制逻辑78指定哪个寄存器是所要执行的下一条指令需要利用的。寄存器控制逻辑78将相应的寄存器输出送到X和Y总线。另外，如图4B所示，在寄存器控制逻辑78的控制下，相应的寄存器R0—R15从Z总线接收信息。

ROM控制器104一旦收到来自Super NES地址总线HA或来自Mario芯片的地址就对那地址进行存取。ROM控制器104将在图13中更详细地示出。从ROM 10存取的信息可以装到超高速缓存RAM 94以便快速地执行指令。ROM控制器104和RAM控制器108都具有在Super NES的和Mario芯片的存取要求间进行仲裁的总线仲裁单元。

如后面会进一步说明的那样，Mario芯片还利用状态寄存器(例如在寄存器块76中或在RAM 6、RAM 8中)，这些寄存器可由Super NES CPU存取，可存储用于标识状态条件的标志，例如0标志、进位标志、正负符号标志、溢出标志、“运行(GO)”标志(1表示Mario芯片正在运行而0表示Mario芯片停止运行)；ROM字节读取正在进行标志(表示寄存器R14被存取过)；各种方式指示标志包

括 ALT1 标志、ALT2 标志、直接低字节和直接高字节标志，和指示源寄存器和目的寄存器已由"WITH"前缀命令设定过的标志，和中断标志。

以图 4A 和图 4B 中的方框图形式表示的 Mario 芯片由 Super NES 调用，使 Mario 芯片每秒钟多次接通和关断来执行任务。起初当 Super NES 开启时，存储在 ROM 10 中游戏程序被引导。注意在由 Super NES 处理器和 Mario 芯片处理器执行游戏程序之前，游戏卡首先被认证。顺便解释一下，按照美国专利 USP 4,799,635 指出的方法，可先将 Super NES CPU 置于复位状态，执行与游戏卡和 Super NES 主控制台相关联的认证处理器的认证程序来进行这类认证。

Mario 芯片一开始处于关断状态。在这一时刻，Super NES 可不受限制地对游戏卡程序 ROM 和游戏卡 RAM 存取。当 Super NES 需要利用 Mario 芯片的处理能力来执行图形操作或数学运算时，Super NES 在游戏卡 RAM（或在规定的 Mario 寄存器）中存储它需要 Mario 芯片处理的相应的数据，并将所要执行的 Mario 程序的地址装入 Mario 芯片的程序计数器。要由 Mario 芯片处理的数据可以是必须旋转和放大或缩小的目标的规定的 X、Y 坐标数据。Mario 芯片能执行会实施算法来控制具有不同个数的子画面或活动目标的背景和前景的程序。利用 Mario 芯片的提高速度的增强硬件和软件，可使这些操作获得高速性能。

应用 Mario 芯片来处理子画面能相当大地扩展整个电视游戏系统的能力。例如，Super NES 每帧只限显示 128 个子画面。随着运用 Super Mario 芯片实际上可显示数百个子画面，并例如使它们旋转。

当 Mario 芯片完成了由 Super NES 所请求的功能，就执行 STOP 指令，产生一中断信号并传输到 Super NES 以表明 Mario 芯

片已完成了它的操作——这转而表明了它已准备好执行下一个任务。

Mario 芯片可用于执行诸如高速乘法运算这类小任务或可用于画一幅满是子画面的画面。在任何一种情况，只要 Super NES 在 RAM 或 ROM 总线正由 Mario 芯片使用的时候不去占用这些总线，Super NES 可自由地与 Mario 芯片并行地进行处理。注意即使 Super NES 交给 Mario 芯片对游戏卡上的 RAM 和 ROM 总线的控制，Super NES 还是可以执行从示于图 2 中的其工作 RAM 32 出来的程序。这样，通过将所要执行的 Super NES 程序从程序 ROM 复制到它的工作 RAM，而与此同时由 Mario 芯片执行程序，整个系统的吞吐量就可得到提高。

示于图 5 中的流程图表示出为起动 Mario 芯片从 ROM 在所要求的地址处读取并执行代码而由主 CPU (例如 Super NES CPU)所执行的"运行 Mario(RUN MARIO)"程序的操作顺序。由图 5 代表的例行程序一般是在将该程序从程序 ROM 10 复制到示于图 2 中的工作 RAM 32 之后由 Super NES CPU 执行。每当要 Mario 芯片进行操作时，就由主 CPU 执行这例行程序。

如方框 125 所示，当执行主 CPU 例行程序 RUN MARIO 时，首先完成包括保留 Super NES 寄存器在内的初始化操作。在初始化步骤期间，该例行程序从程序 ROM 10 复制到主 CPU 的工作 RAM 32 中。

如方框 127 所示，存储着所要执行的 Mario 程序代码的 ROM 10 代码存储区被装进 Mario 芯片的某一寄存器。另外如方框 129 所示，在代码存储区中的实际地址被存储在 Mario 芯片的屏面基址寄存器中。

此后，如方框 131 所示，通过指明将使用 4 种、16 种还是 256 种彩色方式来设定 I/O 输入/输出方式。这些方式对应于主 CPU 所

运行的彩色方式。另外还设定按可显示字符的个数来限定屏面高度的方式。

另外还设定将对ROM和RAM总线的控制交给Mario芯片的方式位。对ROM和RAM总线的控制可分开选定，因而Mario芯片可设定为它对ROM总线、对RAM总线或对两者进行存取的方式。若对ROM和RAM均设定"Mario占用(Marioowner)"方式，则主CPU就不能从ROM或RAM读出或向ROM或RAM写入。注意若主CPU试图在Mario芯片正使用程序ROM总线时对程序ROM存取，Mario芯片有一种向Super NES返回伪地址的机能。向该地址的转移会使Super NES一直被占有直到Mario芯片不再需要对游戏卡ROM总线进行存取。

如方框133所示，在Mario芯片的程序计数器装上存储着Mario程序必须执行的第一条指令的地址之后Mario芯片开始运行。

主CPU然后等待从Mario芯片来的中断信号(方框135)。当接收到一中断信号，Super NES被告知Mario芯片已完成了它的运行并已停止(方框137)。如果未收到这类中断信号，则主CPU继续等待中断(方框135)。在这一期间，Super NES可以通过执行出自示于图2中的其工作RAM 32的程序而与Mario芯片的运行相并行地执行程序代码。

Super NES随后检验状态寄存器(例如在Mario芯片寄存器块76中)以确定表明Mario芯片正在运行的Mario芯片的"GO"标志是否已设定(137)。另外，Mario芯片的状态寄存器中的中断标志被设定以表明Mario芯片是主CPU所收到的中断信号的源。这样，在主CPU接收到中断信号之后(135)，对相应的Mario状态寄存器检测以确定Mario芯片是否是中断源(相对于例如是表明垂直回扫消隐间隔的中断信号)。若Mario芯片已停止运行(137)，则清除对于

RAM 和 ROM 的 Mario 占用方式位，完全由 Super NES 对 ROM 和 RAM 进行存取。Super NES 从该例行程序(141) 退出并返回在进入 RUN Mario 程序前它所执行的程序点。

当 CPU 22 游戏程序使 Mario 芯片进入 ROM Mario 占用方式，它必须主动地停止对 ROM 的存取。任何时候只要 CPU22 因某些原因需对 ROM 存取，它就简单地关闭 ROM Mario 占用方式。当 Mario 芯片再一次需要对 ROM 存取时，Mario 芯片会自动地在那里等待，直到 ROM Mario 占用方式再一次还给它。若它运行来自内部超高速缓存 RAM 的程序就可完全不需要这样做。

若 Mario 芯片处于对于 ROM 的 Mario 占用方式，CPU22 游戏程序不要试图从 ROM 读出任何东西是重要的。当发生某个中断是，例如因垂直回扫消隐，它引起不可屏蔽中断(NMI)，CPU22 就自动地试图从 ROM 读取它的中断向量。这是不希望的，因为 CPU 已明确地告知 Mario 芯片它会避开 ROM，而后来发生中断，它还是要从 ROM 读取。在这种情况，即尽管处于 Mario 占用方式，但 CPU 22 仍要对 ROM 存取的情况下，就会使 Mario 芯片假设这是中断向量请求。

在 ROM Mario 占用方式下，读取中断向量期间，Mario 芯片将把中断向量重新定位到 Super NES 的内部工作 RAM 32 中栈区的底部。例如，如果通常中断向量是 $00：ffec 它就 JUMP(转移)至位置 $00：010c。类似地所有来自 $00：ffeX 的中断向量都使 CPU 22 JUMP 到它们相应的位置 $00：010X。这项技术在 CPU 22 不该对 ROM 10 存取时避免它存取 ROM 10，而是把它转移到 Super NES 的机载 RAM 32 中。注意基于 RAM 的中断向量必须含有向中断处理程序的转移或分支转移。即那里应该存有实际代码而不仅仅是向量地址。当 Mario 芯片不处于 Mario 占用 ROM 的方式时，则仍使用平常的 ROM 中断向量，因而最好继续在这些位置指

向相同的地址，以运行到与基于RAM的中断向量相同的地方。

指令集

Mario芯片指令集为对高速图形和其他处理算法的编程提供了有效的手段。下面对某些指令作简明的说明之后，对不同指令所使用的某些寄存器作出说明.还包括指令集中的指令进行详细的罗列。

指令是8位指令，一般在一个时钟周期内执行。但这些指令可由8位前缀指令改动。Mario芯片指令集包括一独特的寄存器优先代用系统，允许程序员在任何指令之前指定目的端寄存器和两个源端寄存器。没有这些"前缀"的优先代用，指令将只在累加器上运行。这样该指令集是具有无数组合的可变字长的指令集。这里有一些一个字节长、在一个周期内运行的基本指令。通过加上前缀指令，程序员就能扩展这些指令的作用。指令可视程序员的需要为8、16或24位。

Mario处理器利用指令启动执行高速的、机载超高速缓存RAM的程序和对存储器延迟/缓冲的I/O。利用单周期的像素标绘命令，启动采用上述像素标绘硬件的操作，就能有效地进行图形处理。

在列出Mario指令集之前，先在下面说明由处理器在执行指令时设定或存取的各种存储器变换的寄存器。首先列出状态标志寄存器。状态寄存器是16位寄存器，下面标明与该寄存器中16位的各位相关的标志。

状态标志寄存器16位

位	标志	
0	—	备用
1	z	零标志
2	c	进位标志
3	s	符号标志

4	v	溢出标志([位 14 向位 15 进位]XOR [位 15 向进位位进位])
5	g	运行标志：1 Mario 芯片运行 0 停止
6	r	(R14) ROM 字节读取正在进行
7	—	备用

"GO"标志(位 5)设定为"1"状态就表明 Mario 芯片正在运行，而设定为"0"状态就表明 Mario 芯片已停止运行(其结果是产生一中断信号送到 Super NES)。该标志位由 Super NES 处理器检验。位 6 是设定来表明当前正在进行 ROM 字节的读取。列在下面的取(GET)字节指令要等该标志清零表明已完成数据读取后才得以执行。状态寄存器的这些最低有效位可以独立地或与余下的 8 位一起由 Mario 芯片处理器或主 CPU 读出。状态标志寄存器的最高有效位由规定的前缀指令设定，定义指令解释的各种方式。

位	方式	
8	alt1	变更(ADD→ADC，SUB→SBC 等…)
9	alt2	变更(ADD→ADD#，SUB →SUB#等…)
10	il	立即字节低位(在 ih 前进行)
11	ih	立即字节高位(缓冲低位字节直到高位字节就绪)
12	b	设定 SReg 和 DReg，由 WITH 设定
13	—	备用
14	—	备用
15	irq	中断标志

在如上所述的 ALT1 方式时，ADD(加法) 指令会解释为 ADD WITH CARRY (带进位加法) 指令，而 SUB TRACT(减法) 指令会解释为 SUBTRACT WITH CARRY(带进位减法) 指令。指令 ALT1 起动该方式。

ALT2 指令将对 ADD 指令的解释变更为 ADD WITH IMMEDIATE DATA (用立即数据的加法)，而对 SUBTRACT 则变更为 SUBTRACT IMMEDIATE DATA (减去立即数据)。"立即"数据是紧接着指令以字节给出的。注意指令 ALT 3 会把位 8 和位 9 都设定为逻辑"1"电平。位 10 和位 11 根据读立即数据是立即高位字节还是立即低位字节来设定。状态寄存器的位 12 定义"b"方式，这里通过利用前缀指令"WITH"设定源端寄存器和目的端寄存器。状态寄存器的位 15 存储有在 Mario 芯片已停止运行之后设定的 Mario 中断信号。

除了上述的状态寄存器以外，Mario 芯片还包括许多寄存器。如上所述，Mario 芯片包括 16 个如图 4A 和 4B 的寄存器块 76 的说明中所描述的 16 位宽的寄存器。这些寄存器大多数是通过用寄存器并能用作数据和地址的存储。但是如上面所注明的，寄存器 R15 在所有时候却总是用作程序计数器。一般来说寄存器起两方面作用，用于同主 CPU 通信和控制执行程序。另外 Mario 芯片还采用其他寄存器，其功能在下表中给出。

寄存器	专用功能
r0	缺省的 D Reg 和 S Reg
r1	PLOT 指令用的 X 坐标
r2	PLOT 指令用的 Y 坐标
r3	无
r4	LMULT 指令结果的低位字
r5	无
r6	FRMULT 和 LMULT 指令用的乘数字
r7	MERGE (合并)指令用的源 1
r8	MERGE(合并)指令用的源 2
r9	无

r10	无
r11	用于子程序调用的连接寄存器
r12	用于 LOOP(循环)指令的计数
r13	LOOP 指令分支转移的地址
r14	ROM 地址，一经改动就开始从 ROM 读取字节
r15	程序计数器

其他寄存器

8 位 PCBANK	程序代码存储区寄存器
8 位 ROMBANK	程序数据 ROM 存储区寄存器 64K 存储区
8 位 RAMBANK	程序数据 RAM 存储区寄存器 64K 存储区
16 位 SCB	屏面基址
8 位 NBP	位平面数
8 位 SCS	屏面列尺寸选择： 256、320、512、640、1024、1280 (屏面 16 和 20 个字符高，在 2、4 和 8 个位平面中)

Mario 芯片还包括一彩色方式 CMODE 寄存器。在该寄存器中有 4 位在例示性实施例中用以产生下面所述的特殊效果。通过设定 CMODE 寄存器所建立的效果，如下面例子中所阐明的那样，是按照所设定的是 16 还是 256 彩色分辨率方式而有所不同。

CMODE 寄存器各位如下：

CMODE 位 0

标绘彩色 0 位(非透明位)

在 16 彩色方式时：

若位 0=1 和所选择的彩色半字节=0 则不作标绘

在 256 彩色方式且位 3=0 时：

若位 0=1 和彩色字节=0 则不作标绘

在 256 彩色方式且位 3=1 时：

若位 0=1 和彩色低半字节=0 则不作标绘

注意：透明性有效 ON=0

透明性无效 OFF =1

透明性 OFF 的唯一用途是将一区域填以 0

(用来清屏)

CMODE 位 1

晃动位

在 16 彩色方式时晃动(高/低半字节给出两种彩色)

若(xpos XOR ypos AND 1)=0 选择低半字节

若(xpos XOR ypos AND 1)=1 选择高半字节

若透明性有效且所选择的彩色半字节为 0 则不作标绘

在 256 色方式时晃动应当没有效果

CMODE 位 2

高半字节彩色位

在 16 彩色方式或在 256 彩色方式且 CMODE 位 3 被置位时

当本位被置位，COLOUR 命令将彩色寄存器的低半字节设定为源字节的高半字节

(用来析取作为另一子画面的高半字节存储的 16 彩色的子画面)

若彩色寄存器的低半字节为 0 则在透明性有效时不作标绘。

CMODE 位 3

复杂位

仅用于 256 彩色方式。该位被置位时，彩色高半字节就被锁，

COLOUR 命令只改变低半字节。

仅从低半字节计算透明性。

在标准的256彩色方式，如果透明性有效，它是从所有位计算出的

```
;16 彩色方式例
    ibt         r0,$C0
    colour                          ;置彩色$C0
    ibt         r0,%0000            ;置0
    cmode
    ibt         r0,$97
    colour
    plot                            ;标绘彩色$7
    ibt         r0,$30
    colour
    plot                            ;不作标绘，因为彩色
                                    为$0
                                    ;(透明性有效且低半字
                                    节=0)
    ibt         r0,%0001            ;置位1
    cmode
    ibt         r0,$40
    colour
    plot                            ;标绘彩色$0
                                    ;(透明性无效)
    stop
;16 彩色方式且置位2的例
```

```
    ibt         r0,$C0
    colour
                                ;置彩色$C0
;256彩色方式且置位3的例
    ibt         r0,$C0
    colour
                                ;置彩色$C0
    ibt         r0,%1000        ;置位3
    cmode
    ibt         r0,$47
    colour
    plot
                                ;标绘彩色$C7
    ibt         r0,$50
    colour
    plot
                                ;不作标绘，因彩色为
                                $C0
                                ;(透明性有效且低半字
                                节=0)
    ibt         r0,%1001        ;置位3和位1
    cmode
    ibt         r0,$60
    colour
    plot
                                ;标绘彩色$C0
                                ;(透明性无效)
    stop
;256彩色方式且置位3和位2的例
    ibt         r0,$C0
    colour
                                ;置彩色$C0
    ibt         r0,%1100        ;置位3和位2
```

```
cmode
ibt         r0, $74
colour
plot                        ; 标绘彩色 $C7
ibt         r0, $03
colour
plot                        ; 不作标绘，因彩色为
                            $C0
                            ;(透明性有效且低半字
                            节=0)
ibt         r0,%1101        ;置位 3、位 2 和位 1
cmode
ibt         r0 $08
colour
plot                        ; 标绘彩色 $C0
                            ;(透明性无效)
stop
```

许多 Mario 芯片寄存器与特殊功能有关联。如上表中所表明的，若不另外指定，系统就将寄存器 R0 缺省为某一特殊指令所需的目的端寄存器或源端寄存器。寄存器 R0 还可用于 ALU 的累加器。如上所述乘法指令返回一个 32 位的结果。最低有效的 16 位存储在寄存器 R4 中。而寄存器 R6 则结合有符号带小数的乘法指令(FRMULT)和长字节乘法指令(LMULT) 一起被采用。

寄存器 R7 和 R8 用于执行合并指令。该指令取用两个规定的寄存器(即寄存器 R7、R8) 并将它们合并在一起以形成子画面的座标数据。这种座标数据是用于对 ROM 表寻址以便将规定的子画面映像在规定的多边形上。这指令通过组合两寄存器的部分来定义包含

在要映像在多边形上的子画面中的下一个像素的彩色的地址，从而有助于有效地执行质地映像操作。

寄存器R11和R13是用来控制子程序执行的。寄存器R11用作子程序调用的连接寄存器并存储程序计数器的内容加1。寄存器11的内容定义循环结束后须存取的地址。寄存器R12用来存储规定所要执行的循环次数的计数。循环的地址则存储在寄存器R13中。

如上所述，只要寄存器R14的内容被修改，就从ROM在寄存器R14中所存储的地址处读出一字节。在这种方式下，可结合下面提到的GET字节指令实现延迟或缓冲的READ操作。

回到上表中的"其他寄存器(Other Registors)"，程序从其开始执行的程序ROM位置是采用24位地址进行编址。该地址最低有效的16位在程序计数器中得到。而确定程序存储区的最高有效位则存储在程序代码存储区(PC Bank)寄存器。

ROM存储区寄存器(ROM BANK)存储最高有效位，以允许Mario芯片处理器取得存储在ROM 10中的程序数据，这些最高有效位是被附加存储在寄存器R14中的16位ROM地址上。类似地，RAM存储区寄存器(RAMBANK)存储高阶地址位以允许Mario芯片处理器存取RAM中的程序数据。为有效地扩展Mario处理器的寻址范围，可将RAM和ROM的存储区寄存器的内容与Mario芯片的ROM和RAM存取指令一起运用。

屏面基址寄存器(SCB)被用来存储正被建立、旋转、放大或缩小的子画面或目标的虚拟位映像的地址。执行PLOT像素指令时，屏面基址寄存器SCB存储RAM中被存取和被写入的地址。

寄存器NBP被用来存储正在使用的位平面数。它一般表明利用2、4或8个位平面。另外，屏面列向尺寸寄存器SCS被用来按照每列中所包含的字符数确定关于虚拟的位映像的信息。

下面列出的Mario芯片指令集说明指令的助记符和在对相应

的指令译码时所执行的相应的功能。首先在下面对不是不言自明的有关指令的某些功能提出简要的评述。

STOP 指令是在 Mario 芯片已完成其操作时执行的，并用以置"GO"标志为零，同时还产生中断信号给主 CPU。

CACHE 指令用以定义要复制到 Mario 芯片超高速缓存 RAM 中并从该 RAM 执行的那部分程序 ROM。执行 CACHE 指令时，程序计数器的内容被装到超高速缓存基址寄存器并使下面要说明的超高速缓存标记复位。

Mario 芯片包括一系列延迟的分支转移指令，其中接着该转移的指令如下表所示的那样执行。转移去的地址是相对于程序计数器的内容的。指令集包括许多种基于下表中所列出条件的延迟转移。

Mario 芯片包括许多"前缀"指令，即至/ 由/ 自 (to/with/from)。这些前缀指令涉及后续指令的数据往来。例如"T0"前缀为下一指令设定目的端寄存器(DReg)。"FROM"前缀为下一指令设定源端寄存器。而"WITH"前缀则两者都设定。

许多指令在操作码中指定第二源寄存器。若没有前缀指令设定 SReg 和 DReg 它们就被缺省为 R0。在不是前缀指令的各个指令后，SReg 和 DReg 两者均设定为 R0。若 Dreg 被设定为 R15(程序计数器)，因而使下一指令在 R15 中存储其内容时，就会起动延迟一周期的转移。

其它前缀指令在状态寄存器的高字节中设定标志以改变后续指令的操作。所有无前缀指令都使状态字的高字节清零。下面是有关后续指令可以怎样通过前缀指令被修改的例子。

```
lsr             ;r0=r0 向右移 1 位
to r4
lsr             ;r4=r0 向右移 1 位
from r4
lsr             ;r0=r4 向右移 1 位
```

```
alt1
from r6
to r5
add r7          ;r5=r6+r7+进位
alt1
with r3
add r3          ;r3=r3+r3+ 进位 (6502 rol)
```

若在状态寄存器中置上"b"标志，"TO"指令就修改为象操作"MOVE"(赋值)指令一样，TO 指令确定信息传送去的寄存器而 FROM 指令则确定信息源。

STW 指令在缓冲器中存储一特别的字，这样就无须一直等到完成存储操作后才执行后边的指令。在这种方式下，采用比处理器慢的 RAM 并不一定会使处理器速度减慢。

LOOP 指令的执行将使得通用寄存器 R12 内容递减 1。若 R12 的内容非零，则开始向 R13 中所确定的地址转移。

Alt1、Alt2 和 Alt3 是设置状态寄存器中上面提到的标志以按下表中所示的不同的方式解释所执行的指令的前缀指令。

PLOT 指令标识所要标绘的像素的 X 和 Y 屏面坐标，并在与 X 与 Y 坐标(如寄存器 R1 和 R2 中所表示的)相对应的屏面位置上标绘由 COLOR 指令所确定的彩色。PLOT 像素指令包括使 R1 内容的自动递增，以有助于高速标绘水平线而免除一额外的递增指令。

若 Alt1 标志被置位则标绘指令被解释为 READ PIXEL (读出像素)指令(RPIX)。通过执行读出像素指令，在指定屏面位置上的像素彩色被读出，它还可用来将不想要的像素信息从标绘硬件冲掉。

读出像素指令 RPIX 实质上是反过来运用标绘硬件，从字符的矩阵读出以确定在指令中所指定的某一特定像素的颜色。COLOR

指令向彩色硬件提供可由指定的源端寄存器的内容确定的下一个像素的彩色。

"CMODE"指令设定彩色方式并能用于产生如上面提供的例中所展示的不同特殊效果。例如，利用 CMODE 指令可使交替的像素以不同彩色交替变化，形成浓淡变化，从而产生晃动效果。该 CMODE 指令还能用来控制透明性从而使子画面的显示会勾划出背景显示。通过设定如上例子的有关彩色方式的标志来确定透明性。

指令集还包括在旋转多边形的计算中用来确定所要显示的目标的梯度或斜率的小数带符号乘法。

递增指令若与寄存器 R14 一起使用就会起动从 ROM 的读出。GETC 指令会从 ROM 取出所存取的字节并将装入彩色寄存器中。

下表说明按照目前较佳实施例的例示性 Mario 芯片的指令集(包括上面已讨论过的那些指令)。

指令集

十六进制代码	助记符	功能
$00	STOP	使 Mario 芯片停止运行并产生 65816 IRQ g=0
$01	NOP	不操作 1 周期
$02	CACHE	将超高速缓存基址置为 PC 并且使超高速缓存标志复位(只要 PC 不等于当前超高速缓存基址)若超高速缓存基址〈〉r15 则超高速缓存基址=r15，使超高速缓存标志复位
$03	LSR	逻辑右移 DReg=SReg LSR1
$04	ROL	带进位循环左移 DReg=SReg ROL1
$05 nn	BRA sbyte	总是作延迟转移 r15=r15+带符号字节偏移

$ 06 nn	BGE sbyte	若大于或等于则延迟转移 若(s XOR v)=1 则 r15=r15+带符号字节偏移
$ 07 nn	BLT sbyte	若小于则延迟转移 若(s XOR v)=0 则 r15=r15+带符号字节偏移
$ 08 nn	BEQ sbyte	若等于则延迟转移 若 z=1 则 r15=r15+带符号字节偏移
$ 09 nn	BNE sbyte	若不等于则延迟转移 若 z=0 则 r15=r15+带符号字节偏移
$ 0a nn	BPL sbyte	若为正则延迟转移 若 s=0 则 r15=r15+带符号字节偏移
$ 0b nn	BMI sbyte	若为负则延迟转移 若 s=1 则 r15=r15+带符号字节偏移
$ 0c nn	BCC sbyte	若进位清零则延迟转移 若 c=0 则 r15=r15+带符号字节偏移
$ 0d nn	BCS sbyte	若进位置位则延迟转移 若 c=1 则 r15=r15+带符号字节偏移
$ 0e nn	BCC sbyte	若溢出清零则延迟转移 若 v=0 则 r15=r15+带符号字节偏移
$ 0f nn	BVC sbyte	若溢出置位则延迟转移 若 v=1 则 r15=r15+带符号字节偏移

$10-$1f	TO r0…r15	(前缀)设 DReg 为 rn(下一操作码的目的端寄存器) DReg=rn
若 b:	MOVE	rn=SReg (无标志置位)
$20-$2f	WITH r0 … r15	(前缀)设 DReg 和 SReg 为 rn (源和目的和 b 标志) DReg=rn SReg=rn b=1
$30-$3b	STW(rn)	在 rn 中的地址处存储 SReg RAM[rn]=SReg(字低/高被缓冲) (偶数地址上的字正常存储)
若 alt1:	STB(rn)	在 rn 中的地址处存储 SReg 的低字节 RAM[rn]= SReg.l (字节被缓冲)
$3c:	LOOP	使 r12 递减 1 若 r12〈〉0 则延迟转移到 r13 中的地址 r12=r12−1 若 r12〈〉0, 则 r15=r13 (TO/WITH/FROM 不起作用)
$3d	ALT1	(前缀)标志 alt1 置位 alt1=1
$3e	ALT2	(前缀)标志 alt2 置位 alt2=1
$3f	ALT3	(前缀)标志 alt1 & alt2 置位 alt1=1 alt2=1
$40-$4b	LDW(rn)	从 rn 中的地址处取出装到 DReg DReg= RAM[rn](字低/高等待) (偶数地址上的字正常装入)

若 alt1：	LDB(rn)	从 rn 中的地址处取出(无符号字节)装到 DReg
		DReg.h=0
		DReg.l= RAM[rn](字节等待)
$4c	PLOT	在 r1，r2(X、Y)处标绘像素且使 r1 递增(N.B. 并未检验 r1 和 r2 是否在屏面上，但会在 RAM 的任何地方绘制)
		标绘(r1,r2)
		r1=r1+1
若 alt1：	RPIX	读出在 r1、r2(x、y)处像素的彩色
		DReg= 点(r1,r2)
$4d	SWAP	交换字节
		DReg.h=SReg.l
		DReg.l=SReg.h
$4e	COLOUR	设定 PLOT 彩色
		标绘彩色= SReg
若 alt1：	CMODE	设定 PLOT 彩色方式
		标绘彩色方式=SReg
$4f	NOT	DReg = 对 SReg 取反(NOT SReg)
$50-$5f	ADD r0…r15	DReg=SReg+rn
若 alt1：	ADC	DReg=SReg+rn+c
若 alt2：	ADD	DReg=SReg+#n
若 alt1+alt2：	ADC	DReg=SReg+#n+c
$60-$6f	SUB r0…r15	DReg=SReg-rn
若 alt1：	SBC	DReg=Reg-rn-c
若 alt2：	SUB	DReg=SReg-#n
若 alt1+alt2：	CMP	SReg|rn(z,s,c,v)

$70	MERGE	将 r7 和 r8 的高字节合并到 DReg 中 DReg.h=r7.h DReg.l=r8.h 在该结果基础上设定标志： s=b15 OR b7 v=b14 OR b6 OR s c=b 13 OR b5 OR v z=b 12 OR b4 OR c
$71-$7f	AND r1…r15	DReg=SReg AND rn
若 alt1：	BIC	DReg=SReg AND NOT rn
若 alt2：	AND	DReg=SReg AND #n
若 alt1+alt2：	BIC	DReg=SReg AND NOT #n
$80-$8f	MULT r0…r15	DReg=SReg * Rn(带符号的 8×8 位)
若 alt1：	UMULT	DReg=SReg * Rn(不带符号的 8×8 位)
若 alt2：	MULT	DReg=SReg * #n(带符号的 8×8 位)
若 alt1+alt2：	UMULT	DReg=SReg * #n(不带符号的 8×8 位)
$90	SBK	将 SReg 存到上一次使用的 RAM 地址
$91-$94	LINK1…4	将返回地址连接到 r11 r11=r15+1…4
$95	SEX	符号从低字节扩展到字 DReg.[b15—b7]=SReg.[b7] DReg.l=SReg.l
$96	ASR	算术右移 DReg=SReg ASR1

若 alt1：	DIV2	除以 2（带舍入） DReg=SReg ASR1 若 DReg=−1 则 DReg=0
$97	ROR	带进位循环右移 DReg=SReg ROR 1
$98-$9d	JMP r8…r13	转移到 rn 中的地址 r15=rn(延迟转移)
if alt1：	LJMP	长转移到 rn 中的地址 (ROM Bank 来自 SReg) 并使超高速缓存复位 r15=rn（延迟转移）
$9e	LOB	程序 ROM 存储区寄存器= SReg 低字节 DReg. h=0 DReg. l=SReg. l
$9f	FMULT	小数带符号乘法 DReg=(SReg * r6). hw(带符号的 16×16 位乘法) c=(SReg * r6). b15
若 alt1：	LMULT	长字节带符号乘法 DReg=(SReg * r6). hw(带符号的 16×16 位乘法) r4=(SReg * r6). lw c=(SReg * r6) . b15

$a0-$af nn	IBT r0 …r15,sbyte	将符号扩展的字节装到rn中 rn=立即字节(符号扩展的)
若 alt1：	LMS r0…r15,byte	从绝对的偏移字节地址取出装到 rn 中 rn=RAM[byte<<1](字数据)
若 alt2：	SMS r0…r15,byte	将 rn 存储到绝对的偏移字节地址 RAM-[byte<<1]=rn(字数据)
$b0-$bf	FROM r0…r15	(前缀)设 SReg= rn SReg=rn
若 b：	MOVES	DReg=rn (z，s & v (符号低字节)标志)
$c0	HIB	高字节 DReg.h=0 DReg.l= SReg.h
$cl-$cf	OR r1…r15	DReg= SReg OR Rn
若 alt1：	XOR	DReg=SReg XOR Rn
若 alt2：	OR	DReg=SReg OR #n
若 alt1+alt2：	XOR	DReg= SReg XOR #n
$do-$de	INC r0…r14	使 rn 递增 rn=rn+1 (TO/WITH/FROM 不起作用)
$df	GETC	从 ROM 缓冲器取出字节来标绘彩色

若 alt2；	RAMB	RAM 数据存储区寄存器=SReg
若 alt1+alt12；	ROMB	ROM 数据存储区寄存器=SReg
$e0…$ee	DEC r0…r14	使 rn 递减 rn= rn−1 (TO/WITH/FROM 不起作用)
$ef	GETB	从 ROM 缓冲器取出不带符号的字节送到 DReg DReg= ROM 缓冲器字节. 零扩展
若 alt1；	GETBH	从 ROM 缓冲器取出送至 DReg 的高字节 DReg= ROM 缓冲器字节. 与低字节合并 DReg = (SReg & $FF)+(byte〈〈8)(采用 WITH)
若 alt2；	GETBL	从 ROM 缓冲器取出送至 DReg 的低字节 DReg= ROM 缓冲器字节.与高字节合并(采用 WITH)
若 alt1+ alt2；	GETBS	从 ROM 缓冲器，取出带符号字节送至 DReg DReg= ROM 缓冲器字节. 符号扩展
$f0- $ffnnnn	LWT r0…r15,word	将立即字装到 rn 中 rn =立即字(经缓冲的)

若 alt1：	LM r0…r15,word	从绝对的字地址取出装到 rn 中 rn=RAM[字地址](字数据)
若 alt2：	SM r0…r15,word	将 rn 存到绝对的字地址

图 6 至图 17 更为详细地示出图 4A 和图 4B 中用方框图示出的各组成部分。为更清楚地展现本发明的独到特征，那些对本技术领域的技术人员来说是常规或显然的，以及会使这些独到的特征不能被清楚看到的电路细节不在下面的图中示出。

可用作 ALU 单元 50 的例示性算术逻辑单元示于图 6。如图 4A 和图 6 所示 ALU 50 与 X、Y 和 Z 总线相连。因此，Mario 芯片的通用寄存器 R0—R15 与 ALU 相连。

ALU50 经 16 位加法器/ 减法器 152 执行加法和减法功能，ALU50 还包括常规的“AND”逻辑电路 154，“OR”逻辑电路 156，和 EXCLUSIVE OR(异或)”逻辑电路 158。

ALU 还包括常规的移位功能电路，其中任何进位位移位到最高有效位位置而结果则经线 160 送到多路转换器 164。另外，ALU 50 执行常规的字节交换操作，可使载于总线上的最低有效字节和最高有效字节交换，而结果则沿线 162 送到多路转换器 164。X 和 Y 总线如图 6 所示与电路 152,154,156 和 158 相连。

加法器/ 减法器 152，电路 154,156,158 的各自输出，移位输出和交换功能输出与 16 位、6 进 1 “出”的多路转换器 164 相连。根据译出的指令，向目的端总线 Z 输出相应的结果。

加法器/ 减法器 152 除了从 X 总线接收 16 位，还根据送到多路转换器 150 的指令译码器输入接收在总线 Y 送来的信息或指令本身的信息。

ALU50 还包括 CPU 标志发生电路 166。CPU 标志电路 166 产

生零，溢出，符号和进位信号以便于装到电路 166 中的至少一个标志寄存器。CPU 标志可由指令译码电路 60 置位，该电路对指令所产生的进位起动、零起动，符号起动和溢出起动信号进行译码，根据加法器/减法器 152 所确定的相应条件来使标志置位。还可根据输入标志电路 166 的目的端(或结果)总线 Z 的内容使这些标志置位。标志例如可用来触发基于较广的条件范围的条件转移操作。

图 7、8A 和 8B 更为详细地示出图 4A 中的像素标绘电路(52、54、56 和 58)。该电路执行 PLOT 命令，它取出某一特定的 X 坐标和 Y 坐标，并在那些屏面坐标处以由 COLOR 命令装载的彩色寄存器 54 的内容所确定的彩色标绘像素。

Super NES 如上面所注明的那样采用字符映像的显示屏面。标绘硬件用以将像素坐标地址数据变换为字符映像的地址数据。

Super NES 的字符是定义在位平面中的。字符可有 2、4 或 8 位平面以确定 4、16 或 256 种彩色。字符定义的每个字节包括该字符的一个像素行的位平面。这些像素从左到右，从高位到低位被确定。对于 256 彩色方式操作，有 8 RAM 位置需要更新。

像素标绘电路具有一个本地缓冲机构，它包括彩色矩阵 206，该矩阵可存储要显示的某一特定字节的所有各位，因为这些位可能最终都需要更新。位平面计数器 208 与彩色矩阵电路 206 相连。像素坐标从 X 和 Y 总线装载到像素 X 和像素 Y 寄存器 202、204。在本例示性实施例，通用寄存器 R1 和 R2 用作图 7 中示出的标绘 X 寄存器 202 和标绘 Y 寄存器。这些寄存器接收按 PLOT 命令规定要标绘的像素的 X 和 Y 座标。

标绘 X 和标绘 Y 寄存器 202、204 与基于全加器和半加器的字符地址计算电路相连，该电路按地址输出到 2 位置滚桶式移位电路 214，进而与标绘地址寄存器 216 和地址比较器 218 相连。标绘 X 寄存器的三个最低有效位与多路分解器 212 相连进而与位未决寄存器

210 相连。

示于图 8A 中的标绘控制器 200 接收表明已对 PLOT 像素(PLOT)或 READ 像素(RPIX)命令译码的信号以及下述其他控制信号。标绘控制器 200 产生以下述方式使用的标绘电路控制信号。

如上所述，标绘控制电路 200 产生在像素标绘硬件 52 中使用的控制信号。如图 8A 所示，像素控制电路 200 接收从位未决寄存器 210 的输出，该输出是经 AND 门 201 与像素控制电路 200 相连的。若位未决寄存器 210 的所有 8 位都被置位，就通知像素控制逻辑 200 可跳过读出周期并可将彩色矩阵 206 的信息写到 RAM。

像素控制电路 200 还响应 PLOT 命令起动其操作。像素控制逻辑 200 也响应 READ 像素命令 RPIX 起动基本相同的操作，不过新信息并不写到彩色矩阵 206 以输出到 RAM。如上所述，若需要知道屏面上特定像素的彩色就执行 READ 像素命令，该命令也可用来将彩色矩阵 206 的现存信息冲掉。

控制器 200 还接收 RAM 完成控制信号 RAM DONE，它表明已完成 RAM 存取。如上所述，RAM 完成信号还用来使标记彩色矩阵 206 的位平面的位平面计数器 208 递增。标绘控制器 200 还接收从地址比较器 218 来的 PLEQ 信号，这表明地址匹配无需将彩色矩阵 206 的内容写到 RAM，从而表明应继续对当前的彩色矩阵内容更新。该标绘控制器 200 还接收告知标绘控制器 200 有关必须读出和写入多少字节的屏面方式(SCR. MD)控制信号。

标绘控制电路 200 产生使彩色矩阵 206 的内容在它的第二缓冲级缓冲的转储控制信号 DUMP(结合图 7 和图 8B 一起参考)。控制器 200 另外产生使位未决寄存器清零信号 CLRPND 和装载位未决寄存器控制信号 LDPND 并将这些信号送到位未决寄存器 210。另外，控制器 200 产生与结合图 8B 说明的彩色矩阵单元相关的 LDPIX 和 BPR 控制信号。

假定像素标绘硬件未被另外占用，指令译码器对PLOT命令的译码和输入到标绘控制器200的PLOT信号就起动产生装载未决(寄存器)信号LDPND。LDPND信号被送到位未决寄存器210而使数据得以从多路分解器212装到位未决寄存器210中。清未决(寄存器)信号CLRPND是响应表明未决数据已写到RAM去的RAM完成信号RAMDONE而产生的。此后位未决寄存器闲置以供下一像素标绘信息使用。

图8C中所示的是说明由标绘控制器200接收的信号，各种地址和数据信号，其他有关的控制信号和标绘控制器所产生的上述输出控制信号之间的关系的时序图。例示性的地址值、数据值等仅仅是用于说明而示出的。

标绘硬件52如下面那样工作。当标绘控制器200确定标绘硬件52未被占用，示于图4A中的彩色寄存器54的内容被装进8×8彩色矩阵电路206的水平行。彩色矩阵206按行装载而按列读出。彩色寄存器54的内容由COLOR命令更新。任何后续PLOT命令将通过彩色寄存器54这个寄存器将彩色数据装进彩色矩阵。

彩色矩阵中装上彩色寄存器各位的垂直位置是由存储在标绘X寄存器202中的3位最低有效位确定的。这样这标绘地址的三位最低有效位就决定了彩色矩阵206中的所要更新的一行位。

位未决寄存器210是用来记录屏面字符的分节的哪些特定位正被更新。寄存器210包括表明各位已被写到屏面的相关部分的16个寄存器标志。该位未决寄存器210是响应由标绘控制器200所产生的信号LDPND而被装载的并由信号CLRPND清除。

若为更新在相同区域的屏面映像而要执行后续的标绘命令，一给定位的操作就与装入到8×8彩色矩阵206的对应于像素的附加彩色数据一同重复。另一位则利用存于标绘X寄存器202中的标绘地址的最低有效位置设于位未决寄存器210中。特定位是经与标绘

X 寄存器 202 相连的 3 路至 8 路多路分解器装载到位未决寄存器 210 中。若所要更新的像素水平位置在 8 个像素以外，或它处在不同的垂直位置时，则已写入矩阵 206 的数据必须读出到 RAM 6(或 RAM 8)此后彩色矩阵 206 就可自由接收新的彩色数据。在接收到要求写到 RAM 的后续标绘命令之前，彩色矩阵 206 的当前内容一直在像素标绘硬件中例如在彩色矩阵 206 中被缓冲。

当来自彩色矩阵 206 的数据被写到 RAM 6 或 RAM 8 就通过使用示于图 7 中的逻辑门，半加器和全加器进行地址变换计算以将 X、Y 坐标变换成 RAM 地址。实际的地址计算是按照下面给出的注释和例示性代码进行的。这些计算会随着所采用的是 4、16 还是 256 彩色方式有所不同。所给出的例示性计算是针对 256 彩色方式的。

这些 256 彩色字符具有 16 字节的 4 组，各自确定位平面对共有 64 字节。

位映像是通过将独特的字符放在所需屏面区的每个位置上而建立的。当进行与 Super NES 相关的标绘时，最好以列组织字符。

例如(128 个像素高的屏面)

字符号

0	16	32	…	…
	1	17	33	…
	2	18	24	…
	•	•	•	
	•	•	•	
	15	31	47	…

Super NES 并不限于 256 个字符，故而位映像容量主要受到存储器和 DMA 传送时间的约束。例如 Mario 芯片能在 128 和 160 个像素高的屏面上标绘。最大屏面宽度是 32 个字符或 256 个像素。

下面的算法是举例说明如何采用按列组织的虚拟位映像控制像

素标绘。

先对所有的位平面从X坐标的三位最低有效位计算像素掩码。

像素号	掩码
0	%10000000
1	%01000000
	• • •
7	%00000001

接下来利用去除了低3位的Y座标计算沿着一列的下行偏移量，以给出沿一列的字符数，然后乘上字符大小。

屏面彩色	以字节为单位的字符大小
4	16
16	32
256	64

接下来从X座标去除低3位计算字符到顶的偏移，乘以列大小。列大小是列的字符数乘以字符大小。

一般的列大小

彩色	字符高	
	16	20
4	256字节	320字节
16	512字节	640字节
256	1024字节	1280字节

Y坐标的低3位给出沿字符向下的字节偏移。所有偏移与当前位映像指针之和给出含有像素第一位平面的字节的地址。随后的位平面交替地从上一次平面移上1字节和移上15字节。然后，像素各位就可用像素掩码置位或清零。各位平面的位被置位或清零呈该像素所需的存储在彩色寄存器54中的彩色数中相应位的状态。

示范码

```
; 以 65816 代码在 4 个位平面上标绘用于我们的游戏演示
; 程序基本上是表驱动的
; 寄存器 A、X 和 Y 是 16 位
  置彩色
; 取彩色并加倍
lda     Colour
asl     a
tax
;设置位平面 0 和 1 的彩色掩码
lda     mask1 tab, x
sta     mask1
;设置位平面 2 和 3 的彩色掩码
lda     mask2 tab, x
sta     mask 2
rts
标绘
; 取水平和垂直坐标
; 使两者加倍并送到 Y 和 Z 寄存器
lda     plot x1
asl     a
tay                     ;Y 是 X 坐标 * 2
lda     plot y1
asl     a
tax                     ;X 是 Y 坐标 * 2
; 取沿列向下的偏移
lda     pyoftab, x
; 加列偏移初值
```

```
clc
adc     pxoftab,y
;加两倍的缓冲器指示(选择位映像)
clc
adc drawmap
tax
;X 是含有所需要的像素的字从位映象基址的偏移
;Y 是像素的 X 坐标 * 2
;作位平面 0 和 1
lda.l     bitmapbase,x      ;取含有像素字
and       pbittabn,y        ;掩去老像素彩色
sta       pmask
lda       maskl             ;将彩色和像素掩码一起掩去
and       pbittab,y
ora       pmask             ;加入其他像素
sta.l     bitmapbase,x      ;存到位映像
;作位平面 2 和 3
lda.l     bitmapbase+16,x
and       pbittabn,y
sta       pmask
lda       mask2
and       pbittab,y
ora       pmask
sta.l     bitmapbase+16,x
rts
;像素位掩码对的 256 字表
pbittab
```

```
    rept32                          ;num_col
    dw  $8080, $4040, $2020, $1010, $0808, $0404
        $0202, $0101
    endr
    ; 字反向的上表
    pbittabn
    rept 32          ;num_col
    dw $7f7f,-$4040,-$2020,-$1010,-$808,-$404,-$202,-
       $101
    endr
    ; 对位平面 0 和 1 的彩色掩码(彩色 0 至 15)
    mask1 tab
    dw $0000, $00ff, $ff00, $ffff, $0000, $00ff, $ff00,
       $ffff
    dw $0000, $00ff, $ff00, $ffff, $0000, $00ff, $ff00,
       $ffff
    ; 对位平面 2 和 3 的彩色掩码 (彩色 0 至 15)
    mask2 tab
    dw $0000, $0000, $0000, $0000, $00ff, $00ff, $00ff,
       $00ff
    dw $ff00, $ff00, $ff00, $ff00, $ffff, $ffff, $ffff,
       $ffff
col_size equ. Number_char_rows * 8 * Number_bit_planes
              (16)                  (4)
; 对字符列表起始位置的偏移
pxoftab
temp=0
```

```
rept 32                    ;字符列数
dw temp, temp, temp, temp, temp, temp,temp,temp
temp=temp+col_size
endr
;沿列表向下偏移
pyoftab
temp=0
rept 32
dw temp
dw temp+2
dw temp+4
dw temp+6
dw temp+8
dw temp+10
dw temp+12
dw temp+14
temp=temp+32
endr
```

更为仔细地参看图7,确定所要标绘的像素位置的屏面上的X和Y坐标被装载到PLOT X和Y寄存器202和204(这些寄存器实质上可以是寄存器块76中的R1和R2寄存器)。装入到PLOT X寄存器202的标绘地址的3位最低有效位按指定的X和Y坐标确定位平面字节中的哪一位将被写入。累加器R0的内容被装入到由标绘X寄存器202的最低有效位所选定的彩色矩阵206的列。

若标绘X寄存器202是0,则定义该像素的8位的各位中将更新最低有效位。随着标绘X寄存器202为0,3路至8路多路分解器将使位未决寄存器中最低有效位置为逻辑"1"。

既然位未决寄存器 210 的相应位表明无需作修改位未决寄存器 210 被 RAM 控制器用来表明无需从 RAM 写出的间隔。

位未决寄存器 210 用作像素掩码缓冲器以防止从 RAM 覆盖写入新数据。为完成这一功能，如图 7 中所示，位未决寄存器 210 的内容作为输入送到彩色矩阵电路 206。

若 BIT_PENDING 寄存器 210 是 0，则计算像素的屏面地址，并将它装到标绘地址寄存器 216，该字节中的像素位置用来使 BIT_PENDING 寄存器 210 中的相同位置位。若 BIT_PENDING 寄存器 210 是非零，则使 BUSY 标志置位。

若新计算出的地址等于 PLOT_ADDR 寄存器 216 的内容，则新的像素位位置在 BIT_PENDING 寄存器 210 中置位并使 BUSY 标志复位。

若新地址不同于 PLOT_ADDR 寄存器的内容，则采取以下步骤。

步骤 1

若 BIT_PENDING 寄存器 210 包含 FFh 则直接进行步骤 3。

步骤 2

从 RAM 的 PLOT_ADDR＋屏面基址处读取字节送到临时的数据缓冲器 PLOT_BUFF 中。

步骤 3

若 BIT_PEND 寄存器 210 所屏蔽的数据缓冲器中的各位全部等于 PLOT_COLOR 寄存器矩阵的第 0 行，则直接进行步骤 5。

步骤 4

将 PLOT_COLOR 寄存器矩阵的第 0 行写入到由 BIT_PENDING 寄存器启用的 PLOT_BUFF 的所有位。将数据缓冲器内容写回到 RAM 的 PLOT_ADDR 处。

步骤 5

执行相同操作(PLOT_ADDR+1)和执行 PLOT_COLOR 寄存器矩阵的第 1 行。

步骤 6

若为 8 或 256 彩色方式，在(PLOT_ADDR+16)上执行相同的操作和执行 PLOT_COLOR 寄存器矩阵的第 2 行。

继续直到所有彩色位更新

标绘 X 和标绘 Y 寄存器的内容是图 7 中所表示的全加器和半加器电路处理的。在图 7 的方框图中对全加器 FA 和半加器 HA 的配置和有关的逻辑电路进行了简化。地址的计算可如下完成：

```
地址= 屏面基址+2*y[0.2]+
      (y[3..7]+x[3..7]*16+(x[3..7]*4)
          &&scr_ht)
      * 字符大小
```

中间项为

```
                        y7  y6  y5  y4  y3
              x7  x6  x5  x4  x3  0   0
          x7  x6  x5  x4  x3  0   0   0   0
```

```
px9 px8 px7 px6 px5 px4 px3 px2 px1 px0
```

因而例如用 6 个全加器和 1 个半加器产生一 10 位的中间结果 px[0..9]。

为使中间结果移位成为所选定的彩色方式用的正确精度，该结果送到由字符大小值所控制的 12×3 多路转换器。这与 y 的较低位 y[0..2]结合形成 16 位屏面地址。要完成该地址计算，这再与允许将屏面置于 1K 边界上的屏面基址值 scr[9..22]相加。

这地址随后送到根据所选择的 4、16 或 256 彩色分辨率用来将所送到的地址信息输入乘以 1 或 2 或 4 的双位置滚桶式移位器

214。

移位电路214的输出被送到起到RAM地址的缓冲存储器作用的标绘地址寄存器216。由于在执行标绘命令之后，寄存器R1和R2即标绘X和标绘Y寄存器的内容会改变，因而该地址需缓冲。

地址比较器218对作为从移位电路214的输出由标绘硬件确定的新地址与存储在标绘地址寄存器216的旧地址进行比较。若地址不同则该地址必须写到RAM。若存储于地址寄存器216的标绘地址与移位电路214的输出相同，地址比较器218就产生(送到标绘控制器200的)控制信号PLEQ。

回到彩色矩阵206，上面已说明过彩色矩阵206是按列读出的。位平面计数器208与彩色矩阵206相连以确定要读出哪一列。位平面计数器208与RAM控制器88相连，当完成RAM操作，RAM控制器88产生使位平面计数器递增的信号。

彩色矩阵206包括诸如图8B所示的这类单元的阵列。在8×8矩阵206的一个矩阵元中有64个这类单元。当对标绘命令译码时，控制器200将指令控制信号LDPIX送到锁存器220，使该锁存器装有来自彩色寄存器54的彩色数据COL。由控制器200产生控制信号DUMP表明彩色矩阵206中的第一级缓冲已完成和数据需输出给屏面。一旦产生DUMP信号，锁存在锁存器220中的数据就送给门电路226和锁存器228。当DUMP信号被有效地送到门电路226，门电路则将数据送到锁存器228。同时，门224关闭进而阻止来自锁存器228的同相输出端的反馈回路使它保持以前所存储的数据。

当数据从RAM读进来并填到数据间隔时，控制信号BPR就向门222提供零输入且LDRAM信号将处于零状态。在这类情况下，来自RAMD输入的数据输入将经过门电路226送进锁存器228。锁存器228中的数据随后可读出如图7所示经RAM控制器送到RAM数据总线。其他这类单元组合在一起将由X，Y像素标识所

表示的像素数据转换成同 Super NES 字符格式相兼容的字符数据。

在图 9 中详细示出的 RAM 控制器 88 产生有关存取游戏卡 RAM (s)的各种控制信号。游戏卡 RAM (s)必须在 Super NES、Mario 芯片的标绘硬件，和从所执行的 Mario 芯片程序取数据过程之间共用。RAM 控制器 88 的作用在于确保在相应的时刻将相应的地址送给 RAM 地址总线。相应时刻的 RAM 存取信号的产生是由图 10 中更详细示出的仲裁逻辑 310 部分地控制的。

RAM 控制器 88 包括一在来自 RAM 数据插脚经 RAMD 数据总线的输入与指令总线之间多路复用的多路转换器 304。响应从指令译码器 60 来的信号使指令总线或 RAM 数据总线选通，相应的 RAM 输出则置于目的端 Z 总线上。

RAM 控制器 88 还包括一 16 位数据寄存器 300，在从指令译码器 60 所收到信号的控制下保留来存储从 16 位 X 总线或 16 位 Y 总线来的向 RAM 写入的数据。装进数据寄存器 300 的数据被分成低字节和高字节，经多路转换器 302 根据从译码器 60 收到的指令而将低字节或高字节送到 RAM 数据插脚。

RAM 控制器 88 还包括一 20 位地址多路转换器 308。多路转换器 308 对应从仲裁电路 310 收到的控制信号(即从仲裁电路 310 中所产生的代码确认 CACK，数据确认 DACK，或标绘确认 PACK 信号获得的信号)，选择其一地址输入。来自 Super NES 地址总线 HA 的地址信号由多路转换器 308 接收，只要 Mario "占用"状态位置为零就经存储器定时信号发生器 312 送到 RAM 地址总线。通过将信号 RAN 送给还接收 RAM 刷新控制信号 RFSH 的仲裁电路 310 告知仲裁电路 310 Mario 芯片占用 RAM 的状态。RAN 和 RFSH 信号用"或"连在一起以形成图 10 中所示的"暂停"信号。

地址多路转换器 308 还从 16 位多路转换器寄存器 306 接收地址输入。多路转换器寄存器 306 根据由指令译码器 60 所产生的选择

信号接收Y总线的内容或指令总线的内容。多路转换器308还接收作为地址输入的数据存储区寄存器314的输出和图9中所示的程序计数器PC的内容。屏面存储区寄存器316的输出用于形成输入到多路转换器308的标绘地址的最高有效位，而从图7的标绘电路输入最低有效位。屏面存储区寄存器316和数据存储区寄存器314均装上了主数据总线HD来的数据并且可由主CPU寻址。这些寄存器示于图9中，但本身不是一定得配置于RAM控制器88中，而是将它们的内容送给RAM控制器。例如数据存储区寄存器314可以在下述ROM控制器104中而屏面存储区寄存器可以配置在标绘硬件52中。

所要输出的多路转换器308输入信号是如下选择的。若发生代码确认信号CACK，则选择代码存储区和程序计数器PC输入。若发生数据确认信号DACK，则选择数据存储区加多路转换器寄存器输入。若出现标绘确认信号PACK，则选择标绘地址。最后若既无CACK、DACK也无PACK信号出现，就选择主机(即SNES)地址输入。

多路转换器308的20位地址输出送到存储器定时信号发生器312，它则在合适的时刻将这些地址信号送给RAM 6、RAM 8。存储器定时信号发生器312接收仲裁块310中葛莱计数器(gray counter)来的输出。存储器定时信号发生器312对葛莱计数器来的输出译码，并产生经RAM地址总线RAMA对图1中的RAM 6、RAM 8寻址的输出信号。另外一种方式是，定时信号发生器312如图1所示产生包括行地址选通信号RAS，列地址选通信号CAS和写入允许信号WE在内的用于存取RAM 6、RAM 8的控制信号。

存储器定时信号发生器312产生一反馈给仲裁逻辑310表明RAM周期结束的DONE信号。存储器定时信号发生器312还产生一用来将外部RAM来的数据锁存进RAM控制器中的数据锁存器

(未示出)的数据锁存信号 DATLAT。RAM 来的数据随后倒如是经 RAM 数据总线 RAMD_IN 送给 Mario 芯片电路。定时信号发生器 312 来的 RAM A 地址信号输出可送给游戏卡上的任一静态 RAM。若游戏卡中用的是动态 RAM 则产生控制信号 CAS、RAS 和 WE。如上面说明的选配项电阻的设定所表明的，静态或动态 RAM 信号是根据 Mario 芯片的配置而相应地产生的。图 9A 中示出定时信号发生器 312 产生的示范性定时信号和其他有关信号。所示出的示范性地址和数据值仅用于说明。RAM DONE 信号则示于图 8C 中。

在合适的时刻存取 RAM 的信号产生是部分地由仲裁逻辑 310 控制的。如图 10 所示，仲裁逻辑 310 接收有关存储器存取输入的信号 CACHE (超高速缓存)请求 CACHERQ，数据请求 DATRQ 和标绘请求 PLTRQ。这些输入信号每一个分别临时地存储在锁存器 325,327,329 中。若所要执行的 Mario 指令出自 RAM 或 ROM，则通过接收到 CACHE 请求信号 CAC HERQ 来起动处理，该信号就图 10 而言是用来确认正执行的指令不是来自超高速缓存 RAM 因而一定是来自 RAM 或 ROM 的。这样 CACHE 请求信号 CACHERQ 表明不能从 CACHE (超高速缓存)94 取出指令执行。数据请求信号 DATARQ 是作为对需对 RAM 存取的指令(例如装字节、装字指令)译码的结果产生的。另外，仲裁逻辑 310 接收响应标绘命令的译码而由标绘控制器 200 产生的标绘请求信号 PLTRQ。

当 Mario 芯片正在运行和 Mario 占用位被置位时仲裁逻辑 310 才起动(如状态寄存器 SUSPEND (暂停)方式位处于"0"状态所表明的)。在收到和存储 CACHE 请求，数据请求，和标绘请求信号后，锁存器 325,327 和 329 分别产生 CRQ，DRQ 和 PRQ 信号。门 331,333 和 335 接收各自锁存器的同相输出端来的这些信号并为这些信号建立优先级。在这方面，CACHE 请求信号具有最高的优先

级，数据具有第二位高的优先级而标绘请求则具有最低的优先级。CACHE 请求信号被给予最高优先级是因为它表明已试图执行来自 CACHE 指令，但必须从RAM 存取指令。若较高优先级请求已经使其各自的锁存器置位，则门电路 333 和 335 用以确保较低优先级的请求无法使锁存器 339 和 341 置位。只有系统不是处于暂停状态，锁存器 337，339，341 才能置位，因为 SUSPEND（暂停）方式信号被输入到门 331，333 和 335 中的各个门。当 Mario 芯片占用即可自由地存取 RAM 时 SUSPEND 方式信号会处于低逻辑电平状态。若 SUSPEND 被置为 1 锁存器 337，339 和 341 不能被置位，当确认锁存器 337，339 和 341 中的任一个已经为“1”（即一个周期已经在进行）时也不能被置位。门 331、333 和 335 建立 RAM 的优先级。若 CACHE REQUEST（超高速缓存请求）锁存器 337 被置位则数据确认锁存器 339 就不会被置位，若 CACHE 或 DATA 请求锁存器被置位则标绘确认锁存器 341 也不会被置位。

锁存器 337 一由超高速缓存请求信号置位就产生超高速缓存确认信号 CACK，而它一由图 10 中的逻辑电路确立，CACHE 94（或 RAM）就可供使用。若图 10 中的逻辑电路确定 RAM 未被其它方式占用，就会同样地产生数据确认信号 DACK 和标绘请求确认信号 PACK，以确认数据请求和标绘请求。

锁存器 337，339 和 341 的同相输出送到门电路 343，进而经 NOR 门 344 使产生 RAM 存取的定时信号的葛莱计数器 345 复位。本技术领域的技术人员会理解葛莱计数器是每次仅有一个输出位改变的计数器，这可方便地用来控制 RAM 存取时间。

由定时信号发生器 312 产生的 DONE（完成）信号由 NOR 门 344 和锁存器 337，339，341 接收。DONE 信号表明一个 RAM 周期已结束。DONE 信号的产生触发仲裁逻辑 310 中的相应锁存器的清零，使已锁存的请求取消。DONE 信号还送到始发电路例如超高速

缓存控制器 68 或标绘控制器 52，以表明 RAM 存取结束。

按照本发明的另一实施例，Mario 芯片可采用双时钟系统。这样 Mario 芯片处理器无需由例如驱动上面提到的 RAM 控制器电路的相同时钟来驱动。例如 RAM 控制器 88 可由 Super NES 来的 21MHz 时钟信号来驱动而 Mario 芯片处理器可由另一个可变频时钟驱动。在此方式下 Mario 芯片处理器可不限定工作于 21 MHz 时钟频率。

按照这示范性实施例的 Mario 芯片可采用异步态机器控制电路；诸如示于图 11 的执行再同步双时钟接口功能的那种电路。若采用不同的时钟系统而不是工作在另一时钟频率的存储器控制器来实施，则图 11 电路可以用来与 Mario 芯片处理器接口。

示于图 11 中的再同步电路接收与时钟信号 CK 不同步的外来的时钟信号 DIN。不论 DIN 的频率是高于还是低于时钟频率 CK，再同步电路都会由 DIN 产生一与 CK 同步的信号。

如图 12 的说明，图 11 所示电路响应信号 DIN 从状态 010，经过 110；100，101，111 回到起始状态 010。图 11 的再同步电路可用在诸如 ROM 控制器 104 和 RAM 控制器 88 这类接收双时钟信号的任何接口电路。

图 11 所示的电路因由门 F 对锁存器 A 置位故而通过将从其闲置或复位态“010”切换成状态“110”而响应外来的信号 DIN。再同步时钟 CK 一旦变低（也许已经为真），锁存器 B 由门 E 复位形成状态“100”，当时钟再次变高时，由门 A 使锁存器 C 置位形成状态“101”。

锁存器 C 在图 11 中的 Q 从所示的该电路产生输出。当输入信号再次变低，锁存器就再次由门 C 置位形成状态“111”。当时钟信号 CK 在到达状态“111”之后又变低时，锁存器 A 由门 G 复位形成状态 011。此后时钟 CK 再变高并且由门 B 使锁存器 C 复位，回到状

态机器闲置态，其输出就不起作用。

图13更为详细地示出图4B的ROM控制器104。ROM控制器104包括超高速缓存加载器400，它可部分地控制将存储在ROM 10或游戏卡RAM中的当前执行的程序指令载入芯片超高速缓存RAM 94。指令是以16字节的组加载进超高速缓存RAM 94的。当在16字节段当中遇到转移指令，在执行转移之前还是必须继续地填满一完整的16字节段。CACHE加载电路400包括一2位状态机，它通过确保16字节CACHE段的剩下的字节加载进超高速缓存RAM 94来响应对转移指令的译码。超高速缓存加载逻辑状态机的第一状态是闲置状态，若程序执行超出超高速缓存器的范围或程序数据已被加载进超高速缓存，则该闲置状态为真。第二状态表明超高速缓存的加载和从游戏卡ROM或RAM来的指令的执行是同时发生的。第三状态是由对转移指令的译码触发的，该状态处于有效直到16字节的超高速缓存段中的所有字节均已被加载。当执行转移而该转移落到与超高速缓存16字节边界不精确对应的地址时就遇到第四状态，在这种情况下，超高速缓存从该边界的开始位置被填充到16字节段中对应于程序所转移的地址的那部分。

图4B中示出的超高速缓存控制器68产生输入到超高速缓存加载器400的CACHE信号，它表明目前在超高速缓存RAM 94中无法提供所需要的指令。因而必须从ROM读取指令。代码存储区信号标识所要存取的地址的三位最高有效位并且表明所要存取的是程序ROM还是RAM。超高速缓存加载器400还包括在程序执行期间保持与程序计数器PC的各最低有效位相应计数的计数器。该计数器是经超高速缓存加载器的PC输入端加载的。

ROM控制器104中的超高速缓存加载电路400还接收WAIT(等待)和GO控制信号，它们表明Mario处理器因某些原因而处于WAIT状态和Mario芯片在“工作”或“运行”方式。在这种情况下超

高速缓存加载电路400产生一CODEFEFCH（代码读取）控制信号，送给示于图13中的NOR门408，进而送给ROM定时计数器406的清零输入。当超高速缓存加载电路400产生代码读取信号CODE FETCH，就优先于数据读取而启动代码读取，因为在数据读取前必须开始代码读取。图10所示的包括优先级逻辑在内的仲裁电路可用来使所产生的信号被给予比数据读取高的优先级。

当清零信号从ROM定时计数器406取消，就起动一计数周期。ROM定时计数器406用来产生ROM RDY定时信号，它表明在ROM数据插脚可获得ROM数据，该信号是从门电路410输出的。

ROM数据就绪信号ROM RDY门与再同步电路402相连，而该电路例如可包括图11中所说明的再同步电路。在达到与处理器时钟同步之后，产生信号ROM DCK使锁存器404复位和产生DATAFETCH信号表示对寄存器R14的存取触发了数据读取，而这又导致发出EN_R14信号。当ROM定时计数器406已达到一规定计数，确保在ROM数据插脚上可获得数据时就产生DATAFETCH信号。

图13中所示的ROM控制器从下述的输入中的任意一个选择出地址信号的多路转换器414的输出端产生ROM地址。代码存储区寄存器412是从Super NES数据总线HD加载的，以确定从哪一个ROM程序存储区取出并执行该Mario代码。代码存储区寄存器412向多路转换器414提供具有23位的ROM地址的8位。该ROM地址的最低有效位是从程序计数器PC的内容得到的。当正向超高速缓存RAM写入数据时，4位最低有效位是来自于由超高速缓存加载器400所产生的CACHE LOAD信号。另外的多路转换器414地址输入是每当存取寄存器R14时，从Mario通用寄存器R14的内容产生的。

对R14的存取致使数据读取锁存器404产生DATAFETCH信号，该信号用作控制信号使多路转换器414选择其R14输入(和从Super NES数据总线HD加载的数据存储区寄存器416的内容)。该数据存储区寄存器416包含有与R14读取操作有关的数据存储区的最高有效位。

数据读取信号另外送给门408，它将起动由ROM定时计数器406进行的计数，进而经门410产生ROM就绪信号ROM RDY，当ROM RDY信号产生时，就可从ROM数据总线ROM D[7：0]获得数据。

地址多路转换器414还接收从Super NES地址总线HA来的ROM地址。根据送给多路转换器414控制输入端的信号"ROM"的状态将选择Super NES地址总线。"ROM"控制信号向Mario ROM控制器表明Super NES拥有对ROM地址总线的控制。

对转移指令译码后，程序计数器的内容加上由超高速缓存加载器的计数器所产生的四位最低有效位被输入地址多路转换器414。这允许向超高速缓存段加载上在转移被译码前正在加载的16字节的剩余部分。

多路转换器422提供ROM控制器104中从ROM数据插脚ROM D至Mario芯片的目的端总线Z的数据通路。已经由锁存器404产生的DATAFETCH信号和由ROM定时计数器406产生的ROM RDY信号被送给门418而使ROM缓冲器420的加载得以进行。ROM数据总线ROM D[7…0]来的ROM数据被装进ROM缓冲器420。

多路转换器422响应指令代码(诸如通过存取寄存器R14触发的自动进行数据读取的GET B这类指令代码)的译码选取一输入。若对代码读取操作译码，ROM控制器104就会将指令送到图15A中所示的Mario芯片中的指令总线。若对GET B指令译码，存储在

寄存器 420 中被缓冲的字节就被置于 Z 总线上。某些 GET B 指令操作涉及 X 总线上经图 13 中所示的多路转换器的相对应的输入端输入的数据，送给目的端总线 Z 的数据可随后装进 Mario 通用寄存器 76 中的一个。

在图 14 中更为详细地示出超高速缓存控制器 68。超高速缓存控制器 68 包括一标记锁存器 506。标记锁存器 506 包括例如表明指令是否已存储在超高速缓存 RAM 94(为便于说明将它配置于超高速缓存控制器中示出)中的 64 个锁存器。

标记锁存器 506 中的 64 个标志中的每一个都对应于存储在超高速缓存 RAM 94 中的 16 位信息。在从 ROM 或 RAM 取出指令执行的同时超高速缓存 RAM 94 被载上指令。在如上说明的那样执行转移指令时，RAM 94 经与图 13 中示出的 ROM 控制器 104 一起说明的超高速缓存加载器 400 被加上 16 字节段的剩下字节。在加载上这些剩余的字节之前，该整个字节段不能经标记锁存器 506 标志为已加载。

参见门电路 510，当程序计数器已从 0 计数到 15，14 位减法器 502 输出一超出范围的信号(它被反转)当 ROM 控制器输出其 ROM 数据就绪信号 ROMRDY (表明将要输出一字节)，门电路 510 就使标记锁存器 506 在由多路分解器 504 寻址的位置上置位。

当对超高速缓存指令译码；在总线 501 上产生一表明后续指令将从超高速缓存 RAM 存储器 94 执行的控制信号。总线 501 上的控制信号被送给超高速缓存基址寄存器 500 的加载输入端，并用于向超高速缓存基址寄存器 500 装入程序计数器 PC 的 13 位最高有效位。同时如图 14 所示，标记锁存器 506 被清零。

超高速缓存基址寄存器 500 的输出和程序计数器的最高有效位(例如第 3 —15 位)被送给减法器 502，它确定程序计数器 PC 来的地址输入是否在超高速缓存 RAM 94 的范围内。减法器 502 例如输

出其六位最低有效位作为超高速缓存RAM地址的最高有效位，而地址的三位最低有效位则从程序计数器PC送来。

超出范围信号O/RANGE是从减法器502来的进位输出信号而产生的，并被翻转。该翻转的超范围信号高时用来起动对锁存器矩阵506中的一个锁存器的置位。该锁存器置位将取决于经多路分解器504从减法器502来的超高速缓存地址输出，并与超高速缓存RAM 94的16字节段相对应，以表明指令是存储在超高速缓存中与输出的超高速缓存RAM地址相应的地方。标记锁存器506的输出被送给多路转换器512，它将64个标记锁存信号中的一个送到NOR门514，这个锁存信号是根据多路转换器的选择输入确定的，它选定与DEMUX504输出的64条选择线中某一条对应的锁存信号作为输出。输入到NOR门514的另一输入是表明所需要的指令不能在超高速缓存RAM 94中找到因而需从外部读取的超范围信号。

图15示出图4A中示出的ALU控制/指令译码器的方框图。如图15所示，ALU控制器/指令译码器60接收超高速缓存RAM 94，ROM控制器104和RAM控制器88来的指令。这些Mario芯片元件不是ALU/指令译码器的一部分，但仅仅为便于说明才在图15中画出。

多路转换器525从超高速缓存RAM 94，ROM控制器104或RAM控制器88选出指令输出，并将所选择的指令输入到流水线锁存器527。多路转换器525选择基于RAM的指令还是基于ROM的指令，取决于代码存储区寄存器的规定位例如位4的状态。这样根据一装入代码存储区寄存器的地址信息，将对ROM或RAM来的指令进行译码。另外多路转换器525也可根据由超高速缓存器68送来的控制信号CACHE CTL的状态而从超高速缓存RAM 94选择指令，该信号表明所要执行的指令是在超高速缓存RAM 94范围

内，而且某一相应的标记位已如超高速缓存控制器 68 的说明中那样被置位。

当被程序计数器起动信号 PCEN. IL. IH 启动时，流水线锁存器 527 接收多路转换器来的 8 位指令，而这起动信号例如当正由 ROM（或 RAM）读取指令时，是由 ROM 控制器 104(或 RAM 控制器 88)产生的。由于从 RAM 或 ROM 读取指令需花一个处理周期还多，因而指令译码器操作是由各自的 ROM 或 RAM 控制器 104，88 所产生的程序计数器起动信号 PCEN 触发的。

另一方面，若从超高速缓存 RAM 取出指令执行，程序计数器起动信号 PCEN 在所有时刻都是起作用的，并且指令执行是以处理器时钟频率全速执行的。由于 ROM 10 的存取时间比超高速缓存 RAM 94 或游戏卡 RAM 的存取时间慢许多，因而对 ROM 存取时，产生 PCEN 信号的间隔，必须不如相应的超高速缓存 RAM 的，或动态或静态 RAM 的译码起动信号那样频繁。

暂时存在流水线锁存器 527 的指令被输出给图示性地由门电路 537,539 和 541 所代表的常规的指令译码电路，以产生表示操作代码 1,2,…N 的信号。

加载进流水线锁存器 527 的指令还被送给先行控制逻辑 551 。先行控制逻辑 551 用于提供操作代码的译码预告，它将用来从 Mario 芯片寄存器块 76 中选取合适的寄存器。这样，为了在对操作码进行译码之前优化执行的速度，迅速地确定了需被存取的寄存器，以便对该指令所需的数据进行高速存取。

先行控制逻辑响应指令操作码以及各种程序译码控制标志。指令译码电路 60 包括程序控制标志检测器逻辑 543,它响应以前已译码的操作代码而产生 ALT1 和 ALT2 信号，以表明如上所述的相应的前缀指令已经被译码。下面说明的与此有关的 ALT1 PRE 信号也是由标志检测器逻辑 543 产生的。另外产生 IL 和 IH 信号以表明已

对需要立即数的指令进行译码(这里L和H分别指低字节和高字节)。IH和IL标志用来阻止涉及立即数的指令被当作操作码译码。因而也需要非IL信号($\overline{IL}$)和非IH信号($\overline{IH}$)来起动流水线锁存器527。ALT1和ALT2信号如前面说明的那样用来修改后续产生的操作码，并输入到例如在门电路541所示的译码逻辑537,539,541等，按照前面对这些信号的讨论来修改输出操作码。

先行控制逻辑551根据预译码的操作代码和对先前的操作代码(例如前缀代码ALT1或ALT2)译码时所产生的信号而产生寄存器选择信号。例如如程序控制标志检测逻辑543中所示，若ALT1信号由译码逻辑545译码，则产生ALT1 PRE信号，该信号由程序控制标志检测逻辑543输出，并进而经OR门549送给先行逻辑531。该ALT1 PRE信号还使ALT1锁存器547置位。OR门549还将锁存器547来的ALT1信号输出，并将ALT1信号送给译码逻辑537,539,541等。

在图15中示意地表示先行控制逻辑如何产生4个寄存器选择控制位XSEL0，XSEL1，XSEL2和XSEL3。这四个控制位随后送给在图17中结合寄存器控制逻辑76一起说明的多路转换器620和622，该控制逻辑选取16个寄存器中一个的内容输出给X总线供正执行的指令使用。

这样，指令在装进流水线锁存器527前被送给先行译码逻辑单元529，它产生一寄存器选择位XSEL_U0，进而锁存进锁存器535然后作为信号XSEL0输出。锁存器535由程序计数器信号PCEN起动。类似地逻辑电路531产生锁存进锁存器533作为信号XSEL1输出的XSEL_U1。ALT1 PRE信号被送给先行逻辑551中的各种译码逻辑电路529,531等，并被用于确定由寄存器控制逻辑76所选定的合适的寄存器。例如在先行控制电路551中所示，ALT1 PRE信号是送给逻辑电路531的信号之一它产生锁存在锁存器533中进

而输出信号 USEL1 的 USEL_U1。

图 15B 示出用于说明先行逻辑 551 操作的示范性定时信号。图 15B 示出时钟信号 CK，和有关超高速缓存 RAM 数据存取的示范性指令操作码。还示出表明流水线锁存器被加载时，将执行指令译码操作时，产生寄存器选择信号时和将寄存器来的信息加载在目的端 Z 总线上时的定时信号。

如图 15B 所示，超高速缓存 RAM 数据操作码(操作码 1)将在时钟脉冲 CK 上升沿之后的某一时刻变为有效。操作码存储在流水线锁存器 527 中直到例如第二个时钟脉冲的上升沿，这时操作码 2 被锁存进锁存器 527。紧接着收到锁存器 227 来的输出之后，在图 18 中示意地画出的某个时刻指令译码器 60 开始对与操作码 1 相应的指令进行译码。该指令译码的结果，如上所述相应地将控制信号送给诸如 ALU 50，超高速缓存控制器 68 和标绘硬件 52 等这类 Mario 芯片元件。

示于图 15 中的先行电路 551 通过在对操作码 2 译码前的某一时刻产生信号 XSEL_U 开始寄存器选择译码过程。该 XSEL_U0 信号代表被锁存进锁存器 535 之前的译码逻辑 529 的输出。例如在某一时刻由锁存器 535 输出 XSEL—0 信号，使得该指令所需的数据会在指令执行周期中尽早地被存取，以便于尽快地送给相应的总线。

图 16 中示出用来产生有关 Y 和 Z 总线的寄存器选择信号的寄存器控制逻辑 78 的一部分。多路转换器 604 选取 16 个寄存器的哪一个将从 Z 总线被写入，而多路转换器 606 则选取送给 Y 总线的那个寄存器。

多路转换器 604 和 606 分别接收来自 4 位寄存器 600 和 602 的输入。寄存器 600 和 602 用来完成上述的“FROM”和“TO”前缀指令。寄存器 600 和 602 分别由“TO”和“FROM”前缀的译码来起动，

它们将指令总线的最低有效位送给寄存器 600 和 602。寄存器 600 和 602 响应用来使上述的控制标志复位的指令而被清零。

多路转换器 604 和 606 另外接收从寄存器块 76 中的各种寄存器来的输入。另外，多路转换器 604,606 接收从指令总线上的最低有效位来的输入来执行其四个最低有效位确定指令的目的端或源端寄存器的指令。另外为使 Super NES 可对该寄存器存取，将 Super NES 地址总线来的规定的最低有效位送给多路转换器 604 和 606。该多路转换器 604 和 606 分别选取送给 Z 和 Y 总线的寄存器。

图 17 示出寄存器块 76 和配置于图 4B 的寄存器控制逻辑 78 中的附加寄存器选择控制逻辑。由对 FROM 指令译码就产生的 FROMSET 信号设定 FROMX 寄存器 618。一收到 FROMSET 信号，Y 总线的内容就被加载进寄存器 618。装进寄存器 618 的数据随后成为用在后续指令执行中的数据。寄存器 618 的内容被作为输入之一送给多路转换器 622，。多路转换器 622 还接收寄存器 R0 (用于缺省寄存器)的内容作为其输入之一。

多路转换器 622 的另一输入是多路转换器 620 的输出。多路转换器 620 接收程序计数器(即寄存器 R15)的内容、在执行 MERGE 指令时所用的寄存器、和寄存器 R1(例如在执行标绘指令中所用的)来的输入作为输入。多路转换器 620 根据由图 15A 中示出的先行逻辑 551 所产生的 XSEL2 和 XSEL3 位的状态选取这些输入的一个输入。

多路转换器 622 的另一个输入与 Y 总线上的内容相接，以将与 Y 总线上相同数据放在 X 总线上。如前所述，多路转换器 622 的另一输入是上述 FROM X 寄存器 618 的输出。多路转换器 622 的输出是根据图 15A 中所产生的 XSEL0 和 XSEL1 位的状态选取的，并送给 X 总线。

与寄存器 R0—R15 中的许多寄存器有关的专用功能在上面已

详细说明，这里将不重复。寄存器R0—R3的输出送给多路转换器608，寄存器R4—R7的输出送给多路转换器610，寄存器R8—R11的输出送给多路转换器612，而寄存器R12—R15的输出送给多路转换器614。多路转换器608，610，612，614各自的四个输入中的一个均是由YSEL1和YSEL0选取的，这些是从图16中所示的多路转换器606输出的。多路转换器608，610，612和614的输出进而输入到多路转换器616。多路转换器616根据图16中的多路转换器606输出的YSEL2和YSEL3的状态选取四个输入中的一个。多路转换器616使其输出送到缓冲寄存器617，其输出进而送到Y总线。

回到寄存器R0—R15的输入，每个寄存器有一个由如上结合图16所示而产生的ZSEL位0至位3所选取的起动输入。每个寄存器还具有一时钟输入CK和一数据输入DATA—IN，在相应地缓冲之后经该数据输入端从Z总线接收数据。

结合各种乘法操作使用的寄存器R4还包括停用低位和停用高位输入以及允许低位和允许高位输入。寄存器R15即程序计数器PC，它可接收从图13的ROM控制器中的超高速缓存加载器400来的信号CCHLD，在当前的超高速缓存16字节段被装进超高速缓存RAM之前禁止转移操作。程序计数器另外从指令译码器接收一程序循环暂停信号LOOPEN，这表明得进行分支转移并允许向PC加上寄存器R13的内容。寄存器R15另外接收加电复位信号RESET和执行循环指令时将寄存器R13的内容装入程序计数器的输入RN。

如上所述，本发明的图形协处理器结合主电视游戏系统可以方便地用来产生涉及例如基于多边形目标的旋转、放大和/或缩小的各种特殊效果。图18是画出梯形用的示范性Mario芯片程序的流程图，说明可如何对Mario芯片编程以产生所要显示的基于多边形的目标的一部分。下面提出一种用于产生这类多边形的Mario程序同

时对 Mario 硬件如何执行程序加以详尽的解释。

首先参见图 18 中示出的高层流程图，最初将寄存器块 R1 至 R15 的某些寄存器与梯形的产生中所用的变量关联起来(例如寄存器 R1 存储像素 X 位置，寄存器 R2 存储像素 Y 位置线，寄存器 R7 存储梯形高度等)。此后，如方框 650 所示建立循环计数器和计算初始像素值。

如方框 652 所示，随后进行查询以确定梯形水平线之一的长度。若从该线的终点减去该线的起点的结果是负值(—VE)，则该程序分支转移至方框 660。若从该线的终点减去该线的起点的结果是正值表明还未超出该线的长度，则使循环计数器递减(654)并执行标绘像素指令而画出合适的像素(656)。

如方框 658 所示，随后进行查询以确定循环计数器的内容是否为零。若循环计数器不为零则进行转移以分支转移回方框 654 以使循环计数器(654)递减并标绘另一个像素(656)。

若循环计数器等于零，则多边形左侧的 X 坐标和多边形右侧的 X 坐标被更新(660)。此后使梯形 Y HEIGHT (Y 高度)递减(622)，若结果不为零则程序将通过分支转移回方框 650 重新执行(664)，使 Y 坐标递增以移到下一扫描线(665)。若 Y HEIGHT 等于零，则程序将全部执行好，梯形就完成(666)。

为说明产生图形的 Mario 芯片指令集的运用，下面给出实施图 18 流程图绘制梯形的示范性程序。

```
; 绘制梯形循环
rx=1            ; 标绘 x 位置
ry=2            ; 标绘 y 位置
rx1=3           ;顶线左侧 x 位置
rx1inc=4        ; 顶线左侧 x 位置递增
rx2=5           ;顶线右侧 x 位置
```

```
rx2inc=6          ; 顶线右侧 x 位置递增
rdy=7             ;梯形 y 高度
rlen=12           ;循环计数，水平线(hline)长度
rloop=13          ; 循环标记
hlines
    miwt rloop, hlines 2    ; 设定 hline 循环的起始
hlines 1
    mfrom  rx1          ;x=(rx1)〉〉8
    mto    rx
    mhib
    mfrom  rx2
    mhib
    mto    rlen
    msub   rx           ; 长度,rlen=(rx2)〉〉8)—(rx1)〉〉8)
    mbmi-  hlines 3     ; 若 rlen<0 则跳过 hline
    mnop
    minc   rlen         ; 总是绘制一个像素
hlines 2
    mloop
    mplot               ; 绘制水平线
hlines 3
    mwith  rx1          ;rx1+=rx1inc
    madd   rx1inc
    mwith  rx2          ;rx2+=rx2 inc
    madd   rx2 inc
    mdec   rdy          ;rdy—=1
```

```
mbne    hlines      ; 重复 rdy 次数
        1
minc    ry          ; 和下行的下一个 y
```

为说明 Mario 芯片硬件如何用来执行程序，以下的说明是针对上面给出的梯形生成程序的。在执行梯形生成程序前，主计算机系统例如 Super NES 如上面结合图 5 流程图的说明所解释的那样直接向代码存储区寄存器和屏面基址寄存器写入。另外，Super NES 将从 Super NES 地址总线 HA 译码得出的 XEQ 地址的低字节写入 ROM 控制器 104 中的局域寄存器。Super NES 随后将高字节写到 ROM 控制器 104，它与该局域寄存器的内容组合并送到 Z 总线。此后启动寄存器 R15 用作 Mario 芯片的程序计数器。

一检测到上面 Super NES 向 ROM 控制器 104 写入操作的后沿就使 Mario 的“GO”标志置位。若程序计数器减超高速缓存基址寄存器大于超高速缓存容量，或超高速缓存标志与程序计数器的积减去超高速缓存基址寄存器除以 16 等于零，就将程序计数器的内容送给 ROM 10 而起动 ROM 定时计数器(图 13 方框 406)。

一开始，在执行绘制梯形子程序之前，用于梯形循环程序的变量与 Super Mario 寄存器是相关联的，如梯形程序列的初始部分中所示，例如，作为“标绘 X 位置”的“rx”与寄存器 R1 是相关联的，而变量“rloop”则与寄存器 R13 相关联。

分配好这些寄存器之后，梯形程序开始如下执行。当 ROM 控制器 104 中的 ROM 定时计数器达到 5 的计数(近 200 纳秒)，就将所要执行的第一条指令“IWT rloop，hlines 2”从 ROM 数据总线锁存进图 4A 中的流水线寄存器 62。数据同时被写入超高速缓存 RAM 94。在执行指令“IWT rloop，hlines”时，使程序计数器递增。设置“IL”和“IH”标志以表明指令流的下面两字节是立即数据。当

ROM 定时计数器 406 达到 5，就将立即数据(低字节)写入超高速缓冲 RAM 94，并保存在 ROM 控制器 104 中的暂时寄存器。重复 ROM 读取机能，读立即数的高字节与低字节结合并送到 Z 总线。为设置循环计数器，起动寄存器 R13 并在那儿存储 Z 总线的内容。在该例行程序的这一时刻开始，就从存储器读出每条指令直到遇到循环指令为止。

在执行指令"FROM RX1"时，指令代码的最低四位被装进寄存器控制器中的四位"FROM Y"寄存器 602 (见图 16)。另外允许 RX1 (寄存器 R3)来的数据上 Y 总线并将它存进 16 位"FROM X"寄存器 618。在执行"TO RX"指令时，指令代码的最低四位被装进寄存器控制器中的四位"起动 Z"寄存器 600 (见图 16)。

"HIB"指令是通过将"FROM X"寄存器的十六位内容装入 X 总线而执行的。ALU 将 X 总线的高字节放到 Z 总线的低字节上并使 Z 总线上的高字节设为零。这去除 X 位置的小数部分并在寄存器 RX(寄存器 R1)留下第一水平线的起点。

在执行指令"FROM RX2"时，执行如上所述执行"FROM RX1"中相似的操作。该"HIB"指令导致对梯形的顶线右侧 X 坐标进行操作(类似于上面所说的那些)从而将第一水平线的终点留在寄存器 R0(用作累加器的缺省寄存器)。

"RLEN"指令和"SUB RX"指令是通过从线的终点减去该线的起点而被执行的 RLEN(12)=R0—Rx。若结果为负，符号标志将被置位，表明条件假。

"BMI HLINES3"指令是两字节指令，若符号标志置位，第一字节就设立一标志。若条件标志置位，第二字节是转移偏移(这里 R15 等于 R15 加上该指令)。若不是，R15 不被替换而程序继续正常进行。

执行"INC RLEN"指令以使线长度寄存器加 1 以确保至少标绘

一个像素。“LOOP”指令用于进行R12=R12—1 的计算。若R12 不是零，则 R15(程序计数器)装上 13 的内容藉此实行转移。

若此刻的程序在超高速缓存RAM 94 的范围内，则超高速缓存装载电路 400 将检测出该转移，并继续加载超高速缓存 RAM 94，而在这样做时暂停执行程序。结束时，程序计数器加上其新值而从超高速缓存 RAM 94 读取下面的指令。

为执行“标绘”指令，循环/标绘指令配对形成水平线绘制的算法。标绘指令将使由 R1、R2(作为 X 和 Y 坐标)编址的屏面像素设定为图 4A 中示出的“COLOR 寄存器”54 中所设定的彩色。含有该像素的字符的地址是由标绘硬件 52 计算的。该新像素数据保存在字符线缓冲器(彩色矩阵)，直到 Mario 芯片移到一不同的字符位置进行标绘。当所有彩色信息都已复制到彩色矩阵中的双缓冲机构的第二级，就将该信息写入到外部 RAM。

执行“WITH RX1”的“ADD RX1 INC”指令以更新该梯形的左侧 X 坐标。类似地“WITH RX2”和“ADD RX2 INC”用来更新梯形的右侧。“DEC RDY”，“BNE Hlines 1”和“INC RY”指令用来移到下一个 Y 位置(下一扫描行)直到梯形完成。

下面的程序列表说明可如何对 Mario 芯片编程以使 8 位 X，Y 和 Z 点的阵列旋转。该程序说明根据本发明的例示性实施例对图形协处理器编程以执行旋转操作。下面给出该程序的列表：

```
LISTING ROTATE;
; 由寄存器 rmat 1211, rmat 2133, rmat 2322, ramt3231,
rmat0033 中的旋转矩阵旋转 8 位 X,Y,Z 点的矩阵
; 矩阵元是 8 位带符号小数
; 即 127=127/128= 接近 1
; −128=−128/128=−1
; 这些均被压缩存储在每个寄存器内作为 2 个 8 位单元
```

```
rx              =1      ;x
ry              =2      ;y
rz              =3      ;z
rt              =4      ;temp
rmat 1211       =5      ; 矩阵元 11 和 12
rmat 2113       =6      ; 矩阵元 13 和 21
rmat 2322       =7      ; 矩阵元 22 和 23
rmat 3231       =8      ; 矩阵元 31 和 32
rmat 0033       =9      ; 矩阵元 33
routptr         =10     ; 至旋转点缓冲器指针(ptr)

msh—rotpoints8
  miwt      r14,pointsaddr      ;指向所要旋转点的 ROM 指
                                针
  miwt      r12,numpoints       ; 所要旋转的点的数目
  miwt      routptr,m—rotp-     ;指向旋转点缓冲器的 RAM
            nts                 指针
  mcache                        ; 设定超高速缓存地址
  mmove     r13,pc              ;初始循环地址

mmatrotploop
  mto       rx                  ;取 x
  mgetb
  minc      r14
  mfrom     rmat 1211           ;11
  mto       rt
  mmult     rx                  ; m11 * x
  mto       ry                  ;取 y
  mgetb
  minc      r14
  mfrom     rmat 2113           ; 21
  mhib
```

```
mmult    ry              ;m21 * y
mto      rt
madd     rt
mto      rz              ;取 z
mgetb
minc     r14
mfrom    rmat 3231       ;31
mmult    rz              ;m31 * z
madd     rt
madd     r0
mhib
mstb     (routptr)       ;存储旋转后的 x
minc     routptr
mfrom    rmat 1211       ;12
mhib
mto      rt
mmult    rx              ;m12 * x
mfrom    rmat 2322       ;22
mmult    ry              ;m22 * y
mto      rt
madd     rt
mfrom    rmat 3231       ;32
mhib
mmult    rz              ;m32 * z
madd     rt
madd     r0
mhib
mstb     (rout ptr)      ; 存储旋转后的 y
minc     routptr
mfrom    rmat 2113       ;13
mto      rt
mmult    rx              ;m13 * x
```

```
mfrom     rmat 2322          ;23
mhib
mmult     ry                 ;m23 * y
mto       rt
madd      rt
mfrom     rmat 0033          ;33
mmult     rz                 ;m33 * z
madd      rt
madd      r0
mhib
mstb      (routptr)          ; 存储旋转后的 z
mloop
minc      routptr
```

图 19、20 和 21 说明当结合主计算机系统例如 Super NES 使用本发明的可编程图形协处理器时可产生的某些特殊效果。如图 19 所示，描绘出一个目标即直升机的侧视图。该图并不意图准确地反映使用 Mario 芯片所能产生的高品质显示。图 20 和 21 则示出图 19 给出的直升机的放大的和旋转的图像。本发明的图形协处理器可用来产生包括涉及高速旋转和缩放的基于多边形的目标在内的 3D 型(和其他的)特殊效果，而仅最低限度地加重电视游视主处理系统的负担。

本发明已经详细地说明和图释了，应该理解详尽的揭示仅是用作说明和解释。虽然上述实施例被视作较佳实施例，应该理解该领域技术人员可以作出多种变化和改动，而下面的权利要求正是用来覆盖这类在本发明实质和保护范围之内的变化和改动的。

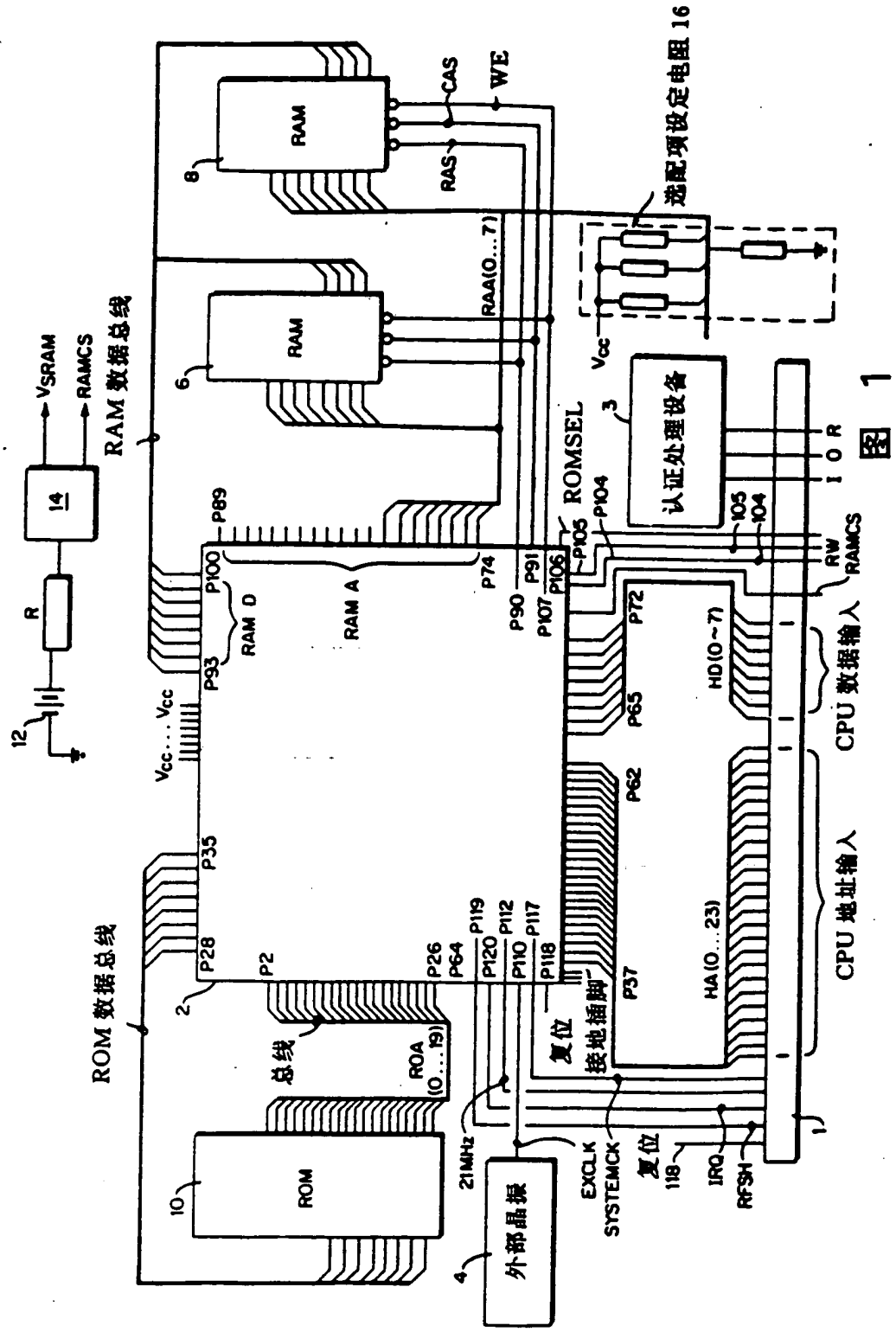
RAM
CAS
WE
RAS
8
6
RAA(0...7)
选配项设定电阻 16
Vcc
3
认证处理设备
ROMSEL
P104
P105
P106
P89
P100
P74
P90
P91
P107
RAM A
RAM D
P93
P72
P65
HD(0~7)
105
104
IOR
RW
RAMCS
CPU 数据输入
Vcc....Vcc
P35
P28
P2
P62
P37
HA(0...23)
CPU 地址输入
P26
P64
P119
P120
P112
P110
P117
P118
2
总线
ROA
(0....19)
复位
接地插脚
ROM 数据总线
ROM
10
21MHz
外部晶振
4
EXCLK
SYSTEMCK
复位
118
IRQ
RFSH
1
VSRAM
RAMCS
RAM 数据总线
14
R
12

图 1

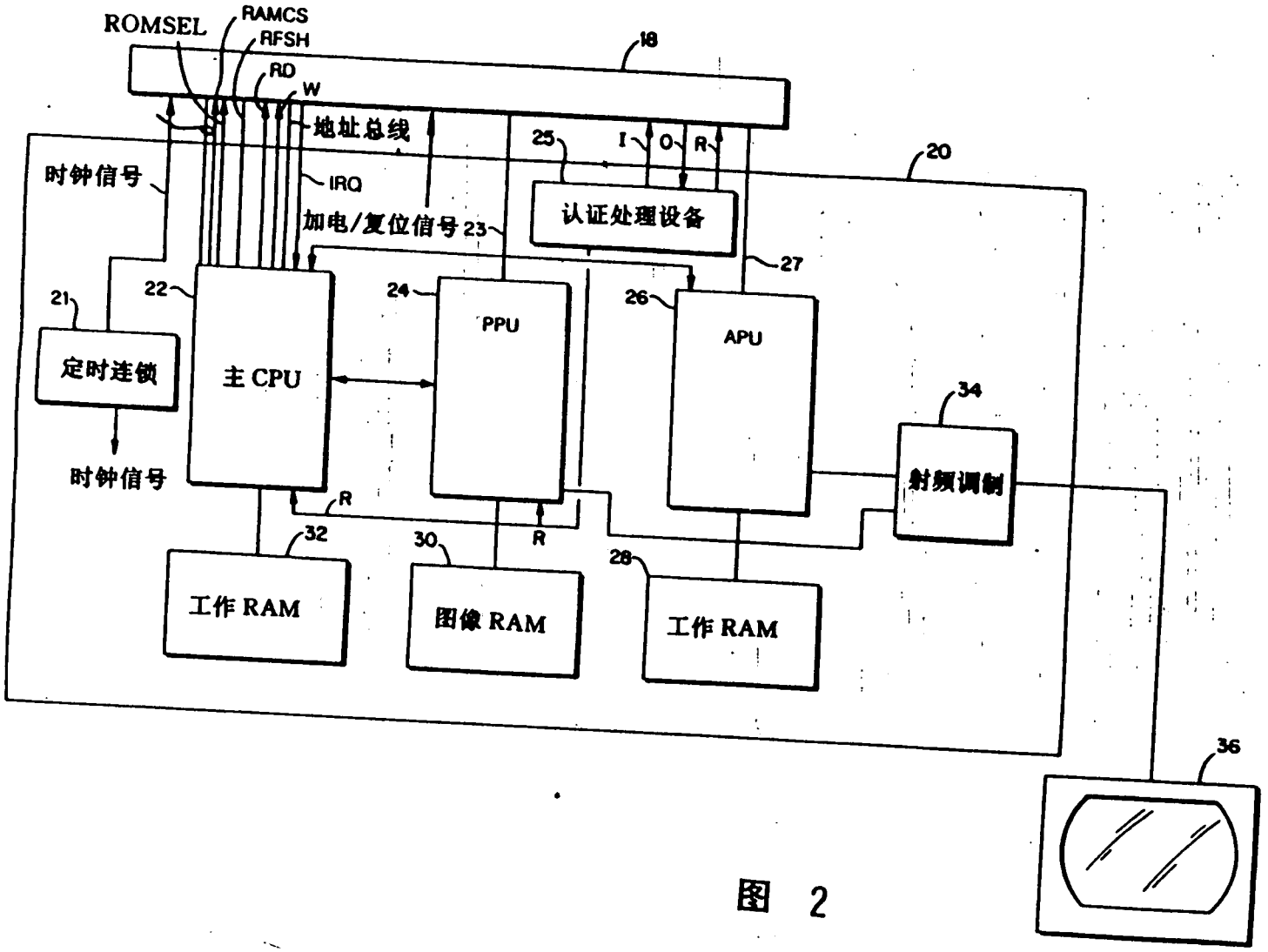
ROMSEL
RAMCS
RFSH
RD
W
18
地址总线
25
I
O
R
20
时钟信号
IRQ
加电/复位信号 23
认证处理设备
27
21
22
24
26
PPU
APU
定时连锁
主 CPU
34
射频调制
时钟信号
R
R
32
30
28
工作 RAM
图像 RAM
工作 RAM
36

图 2

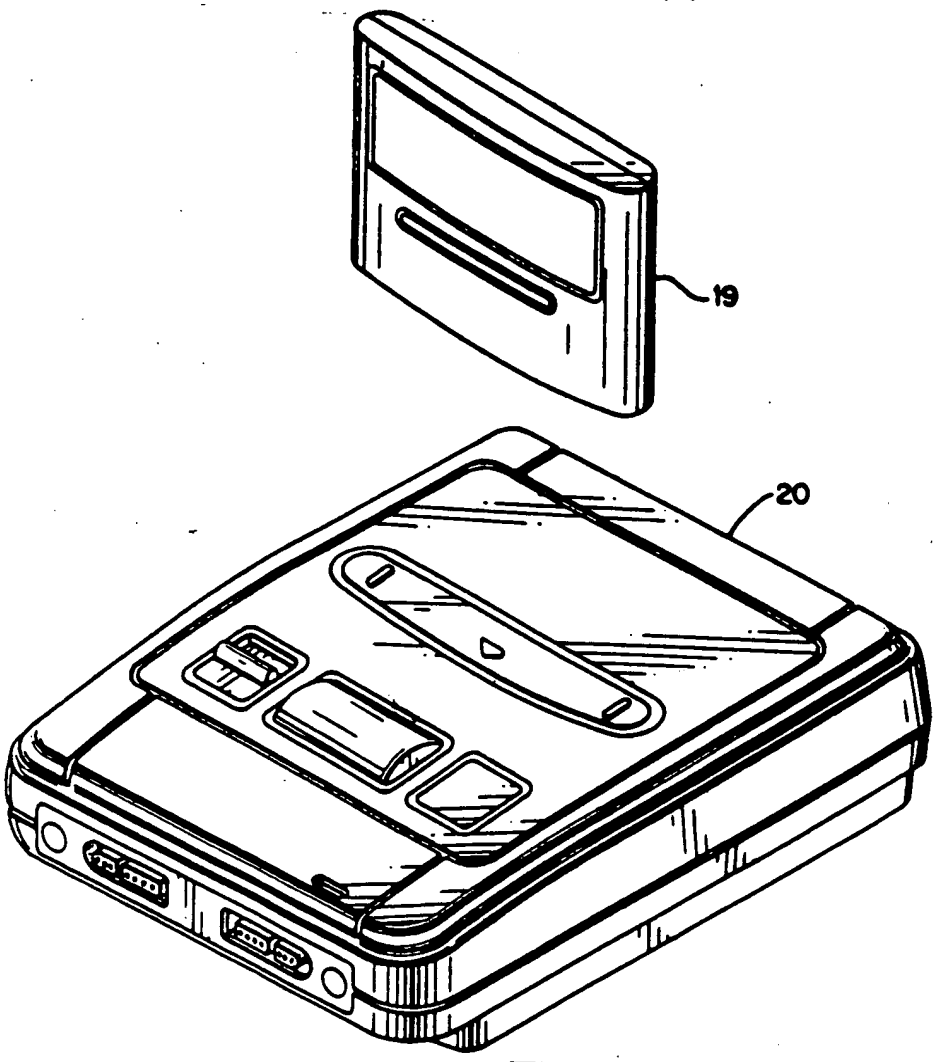

图 3

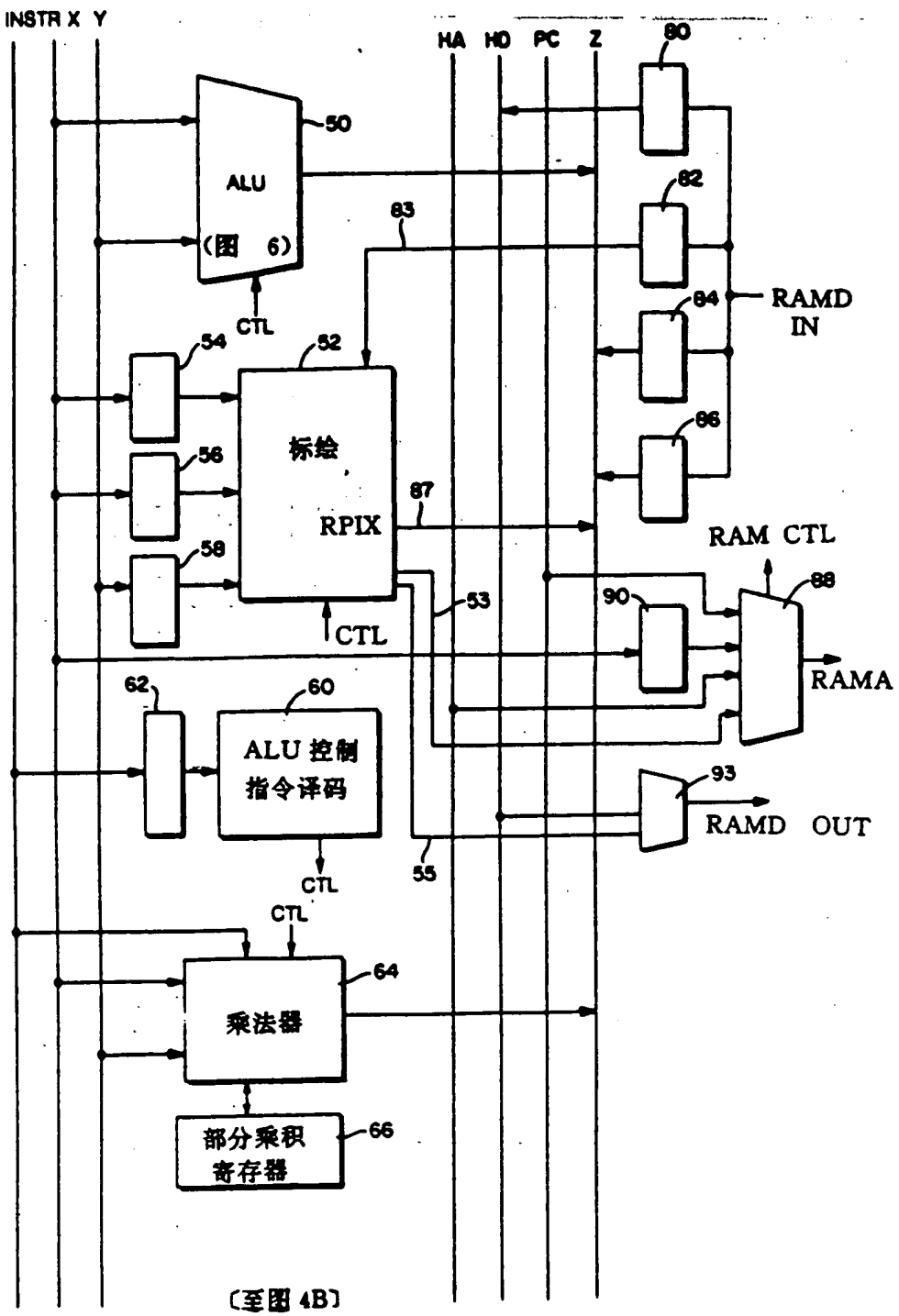
INSTR X Y
HA HO PC Z
80
50
ALU
(图 6)
CTL
82
83
84
RAMD
IN
54
52
标绘
RPIX
56
86
87
58
RAM CTL
88
90
53
CTL
RAMA
62
60
ALU 控制
指令译码
93
RAMD OUT
CTL
55
CTL
64
乘法器
部分乘积
寄存器
66
〔至图 4B〕

图 4A

〔接图 4A〕

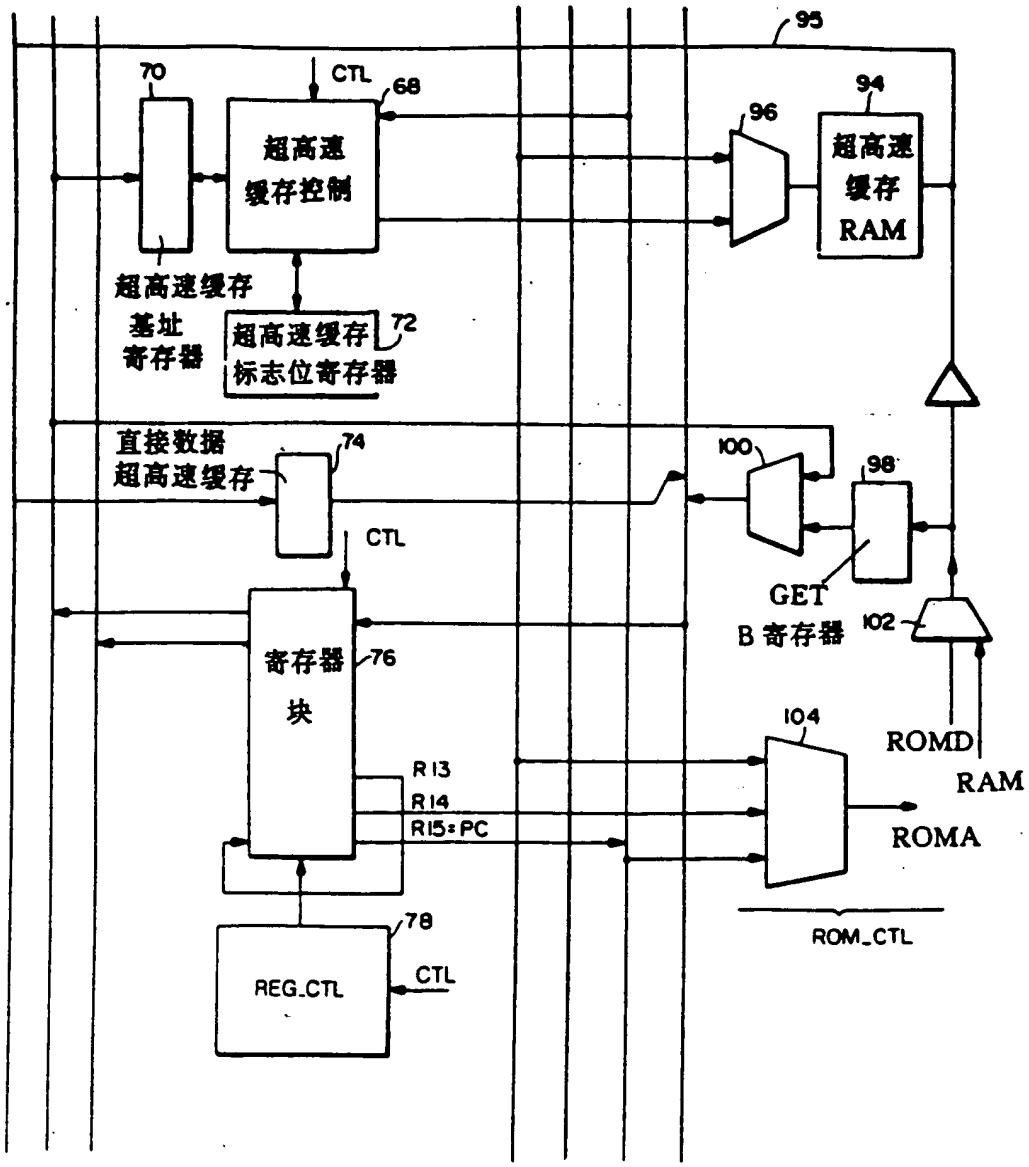
95
70
CTL
68
超高速
缓存控制
96
94
超高速
缓存
RAM
超高速缓存
基址
寄存器
超高速缓存
标志位寄存器
72
直接数据
超高速缓存
74
100
98
CTL
GET
B 寄存器
102
寄存器
块
76
104
ROMD
RAM
R13
R14
R15=PC
ROMA
78
REG_CTL
CTL
ROM_CTL

图 4B

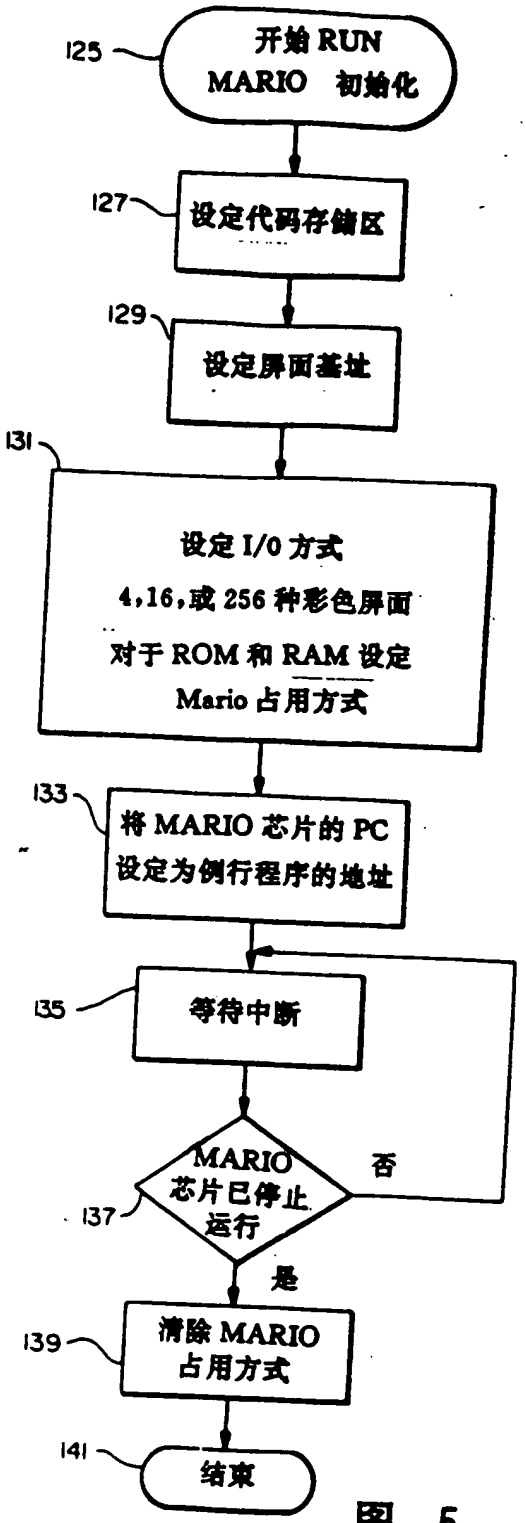
125
开始 RUN
MARIO 初始化
127
设定代码存储区
129
设定屏面基址
131
设定 I/O 方式
4,16,或 256 种彩色屏面
对于 ROM 和 RAM 设定
Mario 占用方式
133
将 MARIO 芯片的 PC
设定为例行程序的地址
135
等待中断
MARIO
芯片已停止
运行
137
否
是
139
清除 MARIO
占用方式
141
结束

图 5

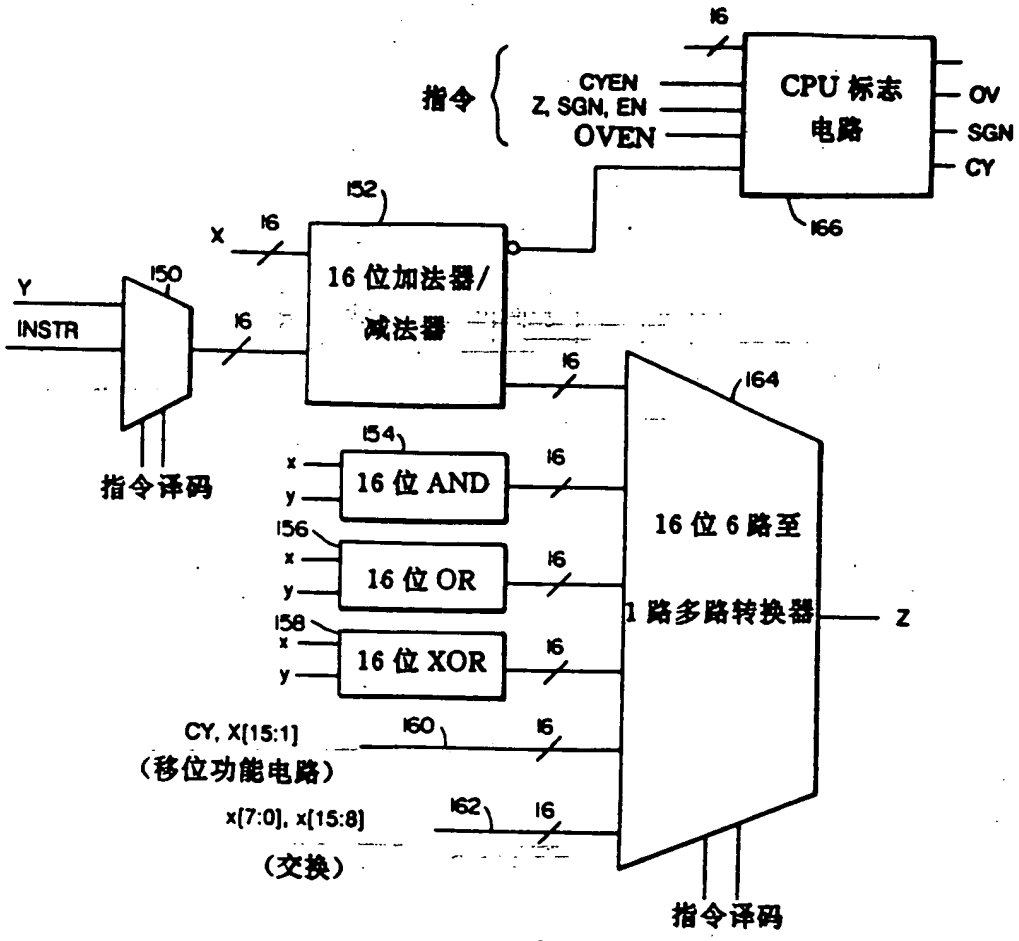
16
指令
CYEN
Z, SGN, EN
OVEN
CPU 标志
电路
OV
SGN
CY
166
152
X
16
16 位加法器/
减法器
Y
INSTR
150
16
指令译码
16
164
154
x
y
16 位 AND
16
16 位 6 路至
1 路多路转换器
Z
156
16 位 OR
158
16 位 XOR
CY, X[15:1]
（移位功能电路）
160
x[7:0], x[15:8]
（交换）
162
指令译码

图 6

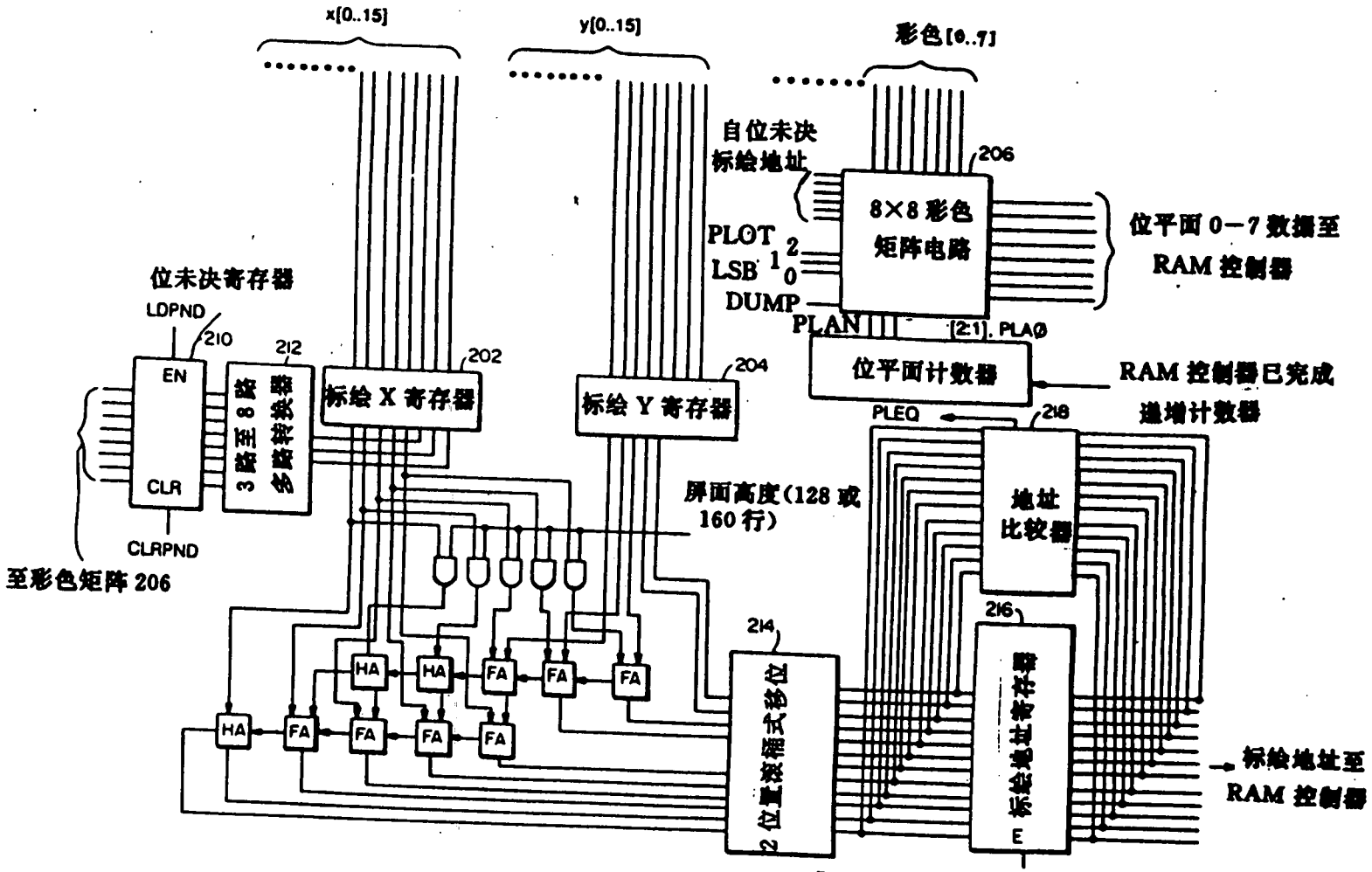
x[0..15]
y[0..15]
彩色[0..7]
自位未决
标绘地址
206
8×8 彩色
矩阵电路
位平面 0—7 数据至
RAM 控制器
PLOT
LSB 2
1
0
DUMP
PLAN
[2:1], PLA0
位平面计数器
RAM 控制器已完成
递增计数器
PLEQ
218
地址
比较器
位未决寄存器
LDPND
210
212
EN
CLR
CLRPND
至彩色矩阵 206
3 输入 8 输入多输入与非门
202
标绘 X 寄存器
204
标绘 Y 寄存器
屏面高度(128 或
160 行)
214
2 位直接型式移位
216
标绘地址寄存器
E
标绘地址至
RAM 控制器
标绘命令
HA
FA

图 7

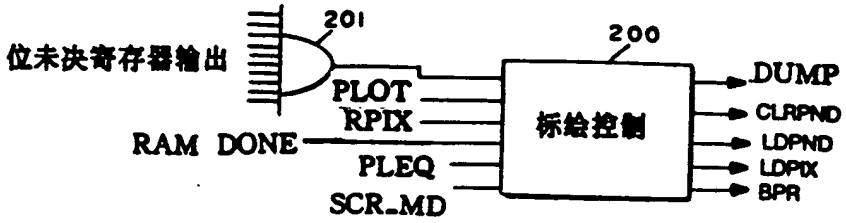
位未决寄存器输出
201
200
PLOT
RPIX
RAM DONE
PLEQ
SCR_MD
标绘控制
DUMP
CLRPND
LDPND
LDPIX
BPR

图 8A

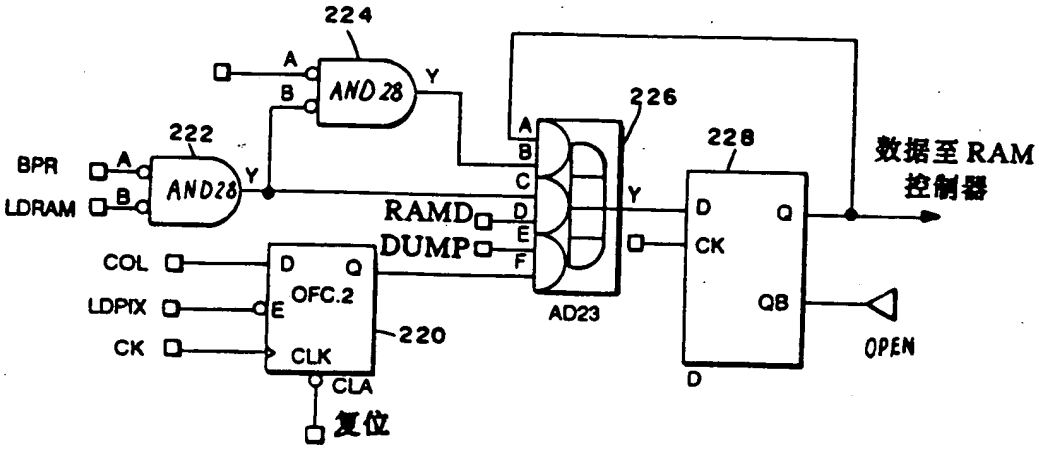
224
AND28
222
AND28
BPR
LDRAM
COL
LDPIX
CK
OFC.2
220
CLA
复位
RAMD
DUMP
226
AD23
228
数据至 RAM
控制器
OPEN

图 8B

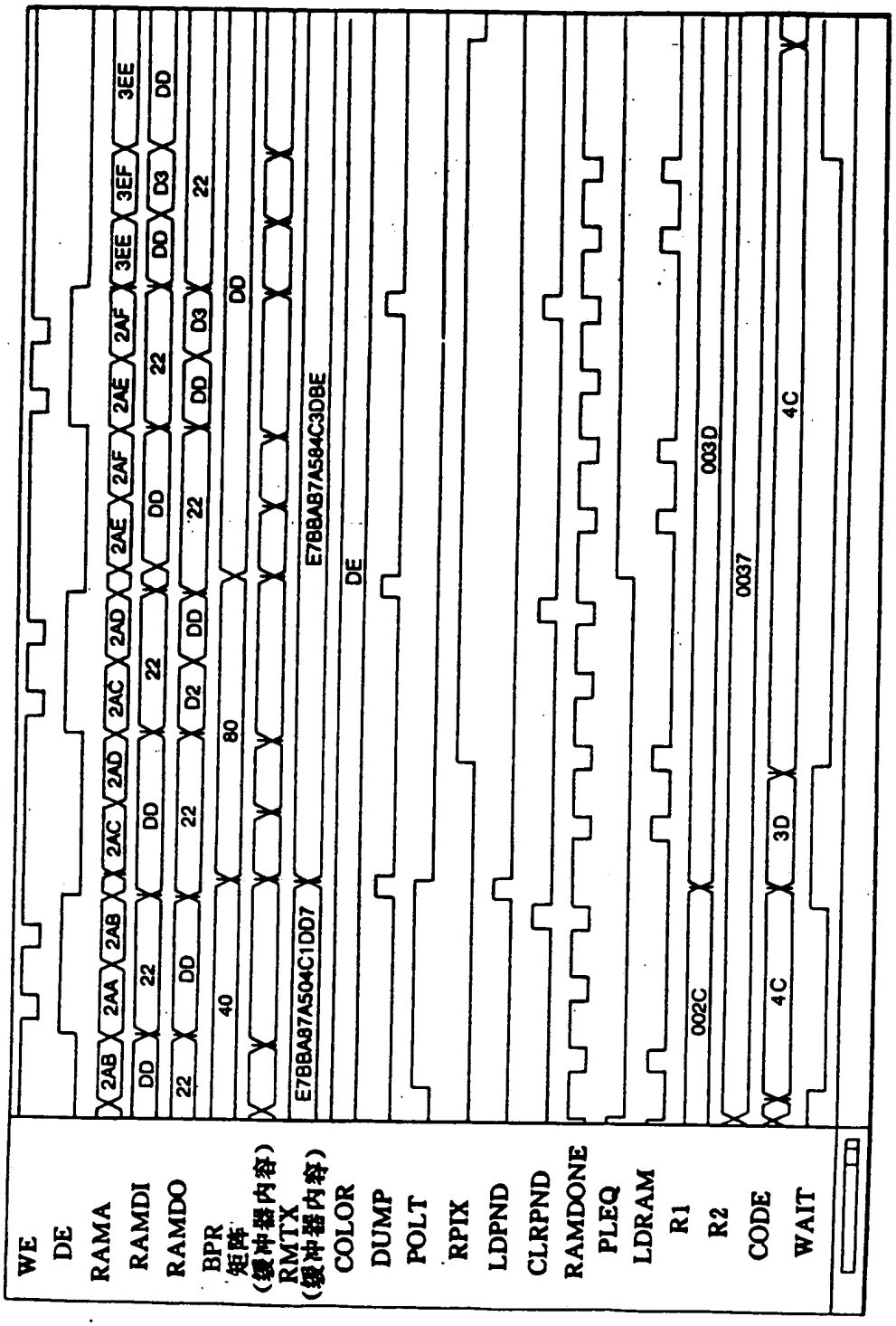
WE
DE
RAMA
RAMDI
RAMDO
BPR
矩阵
(缓冲器内容)
RMTX
(缓冲器内容)
COLOR
DUMP
POLT
RPIX
LDPND
CLRPND
RAMDONE
PLEQ
LDRAM
R1
R2
CODE
WAIT
2AB
2AA
2AB
2AC
2AD
2AC
2AD
2AE
2AF
2AE
2AF
3EE
3EF
3EE
DD
22
DD
22
DD
22
DD
22
DD
22
DD
D3
DD
22
D2
DD
22
DD
D3
DD
80
DD
E7BBA87A504C1DD7
E7BBAB7A584C3D8E
DE
002C
003D
0037
4C
3D
4C
4C

图 8C

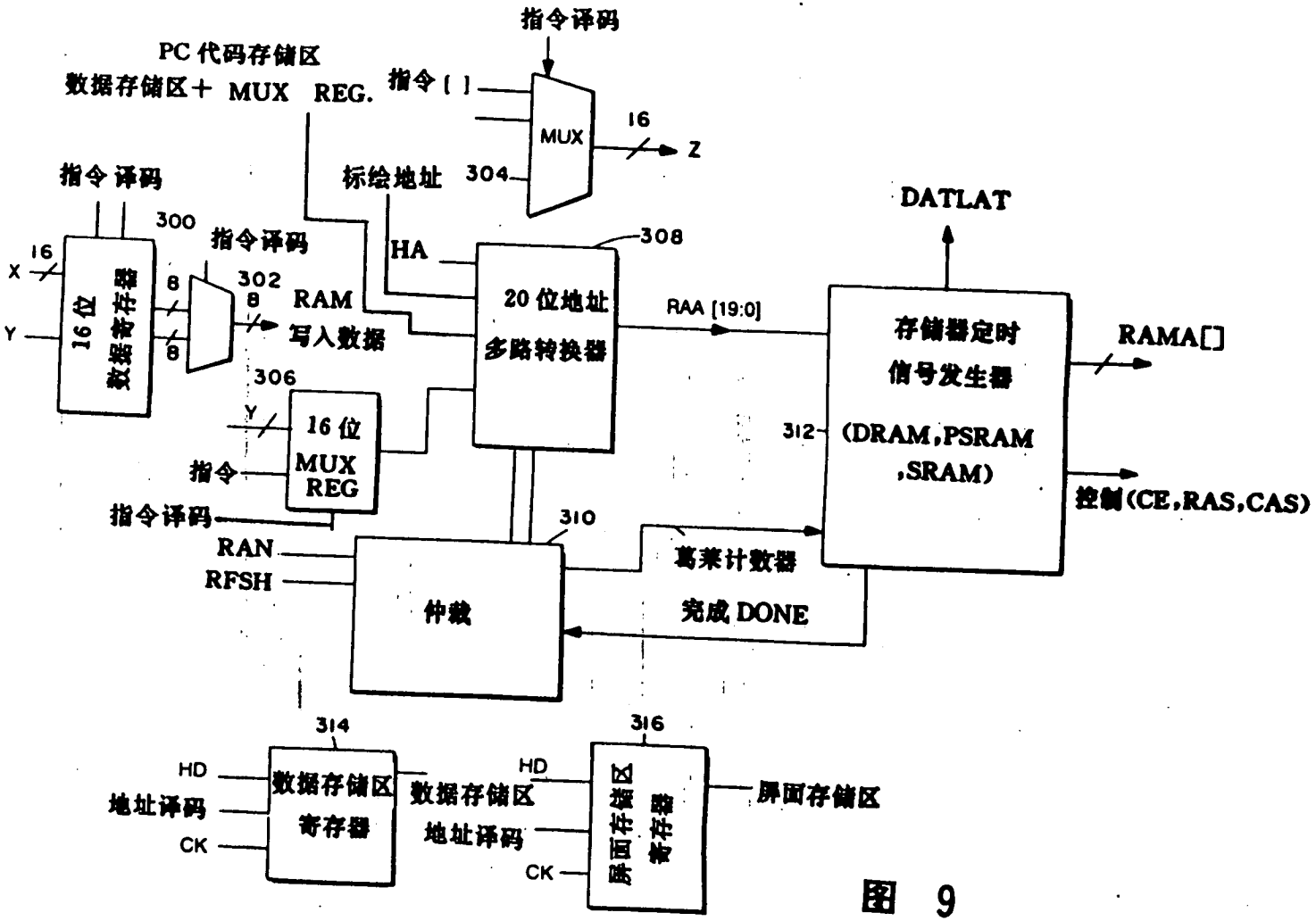
指令译码
PC 代码存储区
数据存储区＋ MUX REG.
指令[ ]
MUX
16
Z
指令 译码
标绘地址
304
DATLAT
300
指令译码
HA
308
16
X
16 位 数据暂存器
8
302
8
RAM
写入数据
Y
8
20 位地址 多路转换器
RAA [19:0]
存储器定时 信号发生器 (DRAM,PSRAM ,SRAM)
RAMA[]
306
Y
16 位 MUX REG
312
指令
控制(CE,RAS,CAS)
指令译码
310
RAN
RFSH
仲裁
葛莱计数器
完成 DONE
314
316
HD
数据存储区 寄存器
地址译码
CK
HD
数据存储区
地址译码
CK
屏面存储区 暂存器
屏面存储区

图 9

102

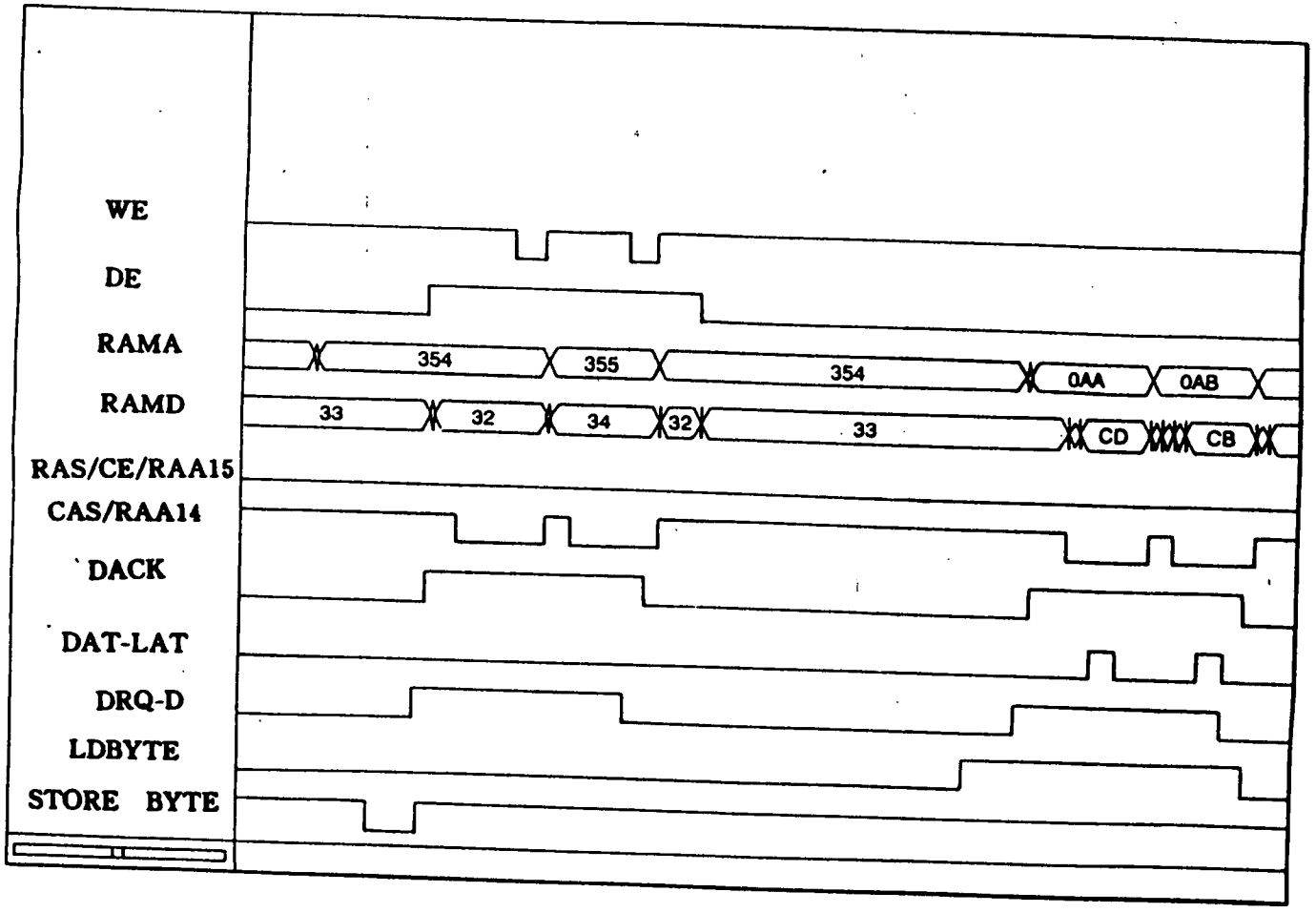
WE
DE
RAMA
RAMD
RAS/CE/RAA15
CAS/RAA14
DACK
DAT-LAT
DRQ-D
LDBYTE
STORE BYTE
354
355
354
0AA
0AB
33
32
34
32
33
CD
CB

图 9A

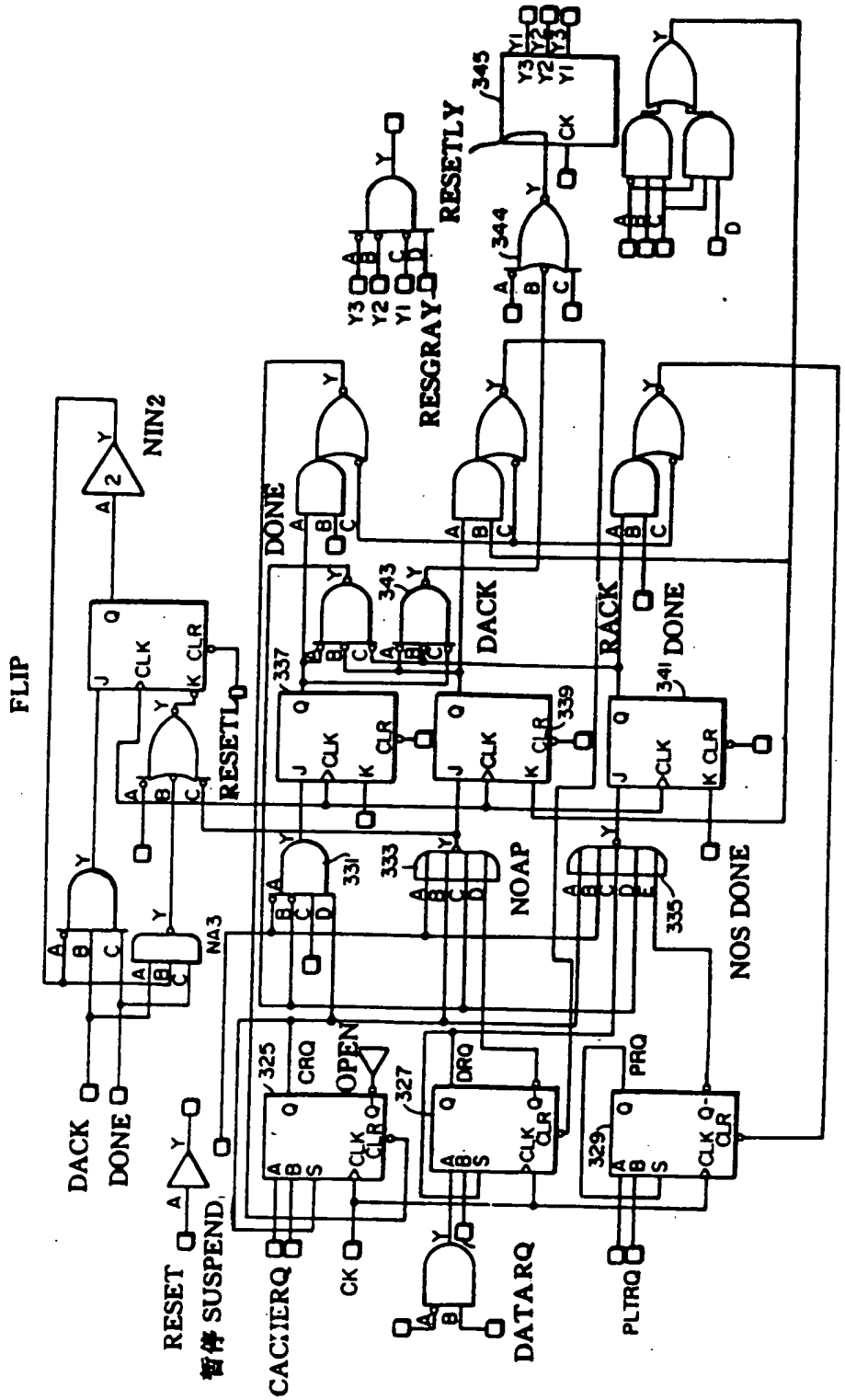
FLIP
NIN2
RESETL
DONE
RESGRAY
RESETLY
DACK
RACK
DONE
NOAP
NOS DONE
NA3
OPEN
CRQ
DRQ
PRQ
RESET
暂停 SUSPEND
CACIERQ
CK
DATARQ
PLTRQ
DACK
DONE
325
327
329
331
333
335
337
339
341
343
344
345

图 10

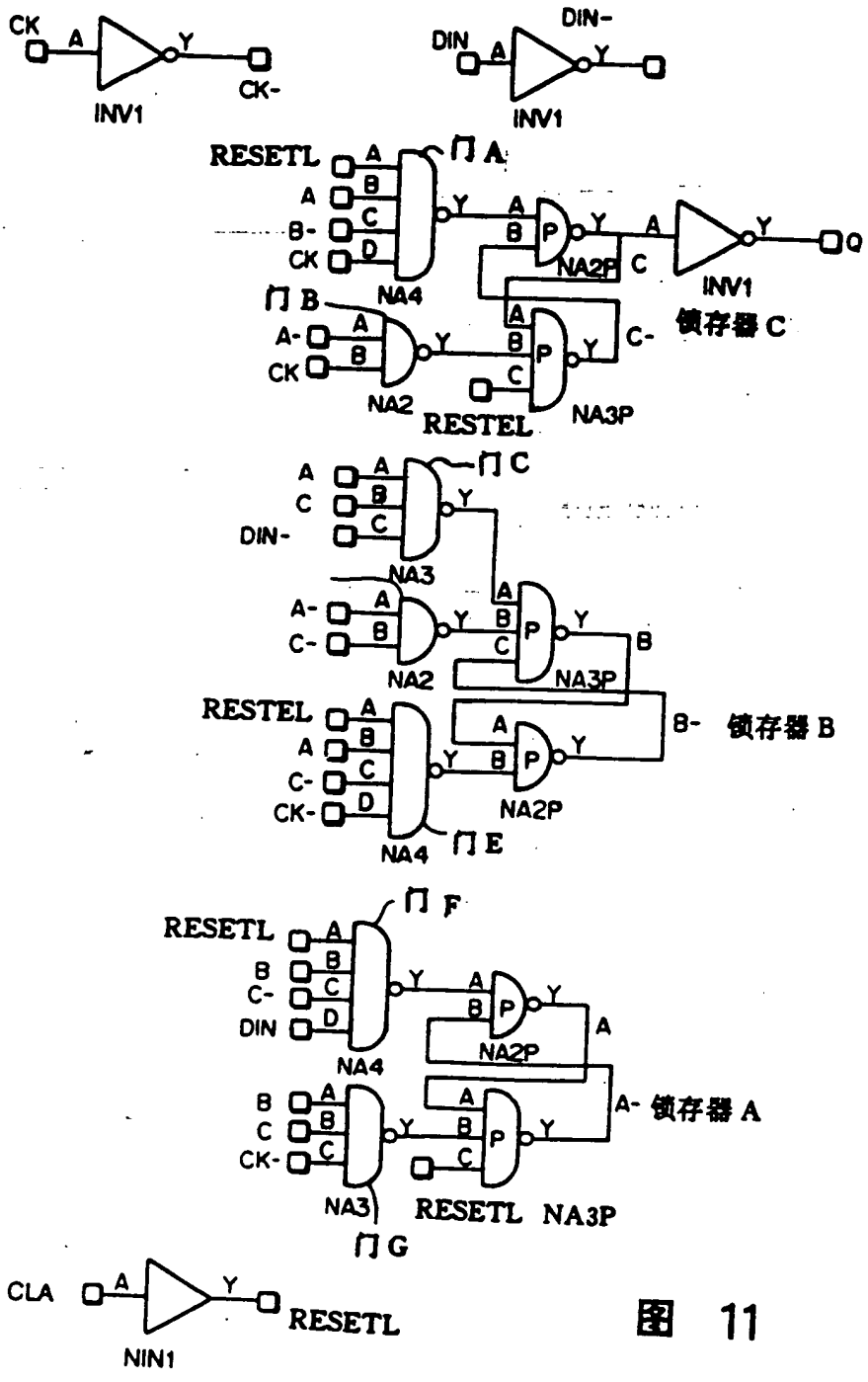
CK
A
Y
CK-
INV1
DIN
DIN-
INV1
RESETL
门 A
A
B
C
D
NA4
B-
CK
Y
A
B
P
NA2P
C
INV1
Q
锁存器 C
门 B
A-
CK
NA2
RESTEL
C-
NA3P
门 C
DIN-
NA3
A-
C-
NA2
B
NA3P
RESTEL
B-
锁存器 B
C-
CK-
NA2P
NA4
门 E
门 F
B
C-
DIN
A-
锁存器 A
CK-
NA3
RESETL
NA3P
门 G
CLA
NIN1
RESETL

图 11

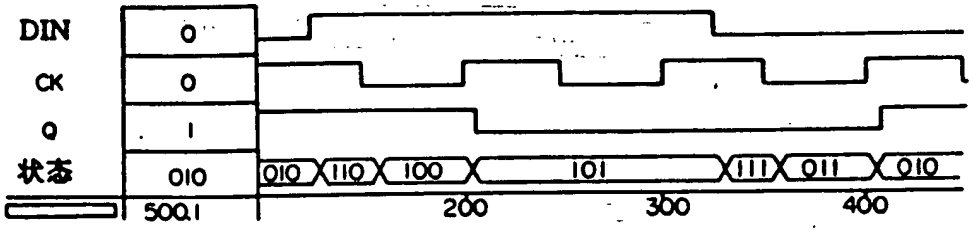

图 12

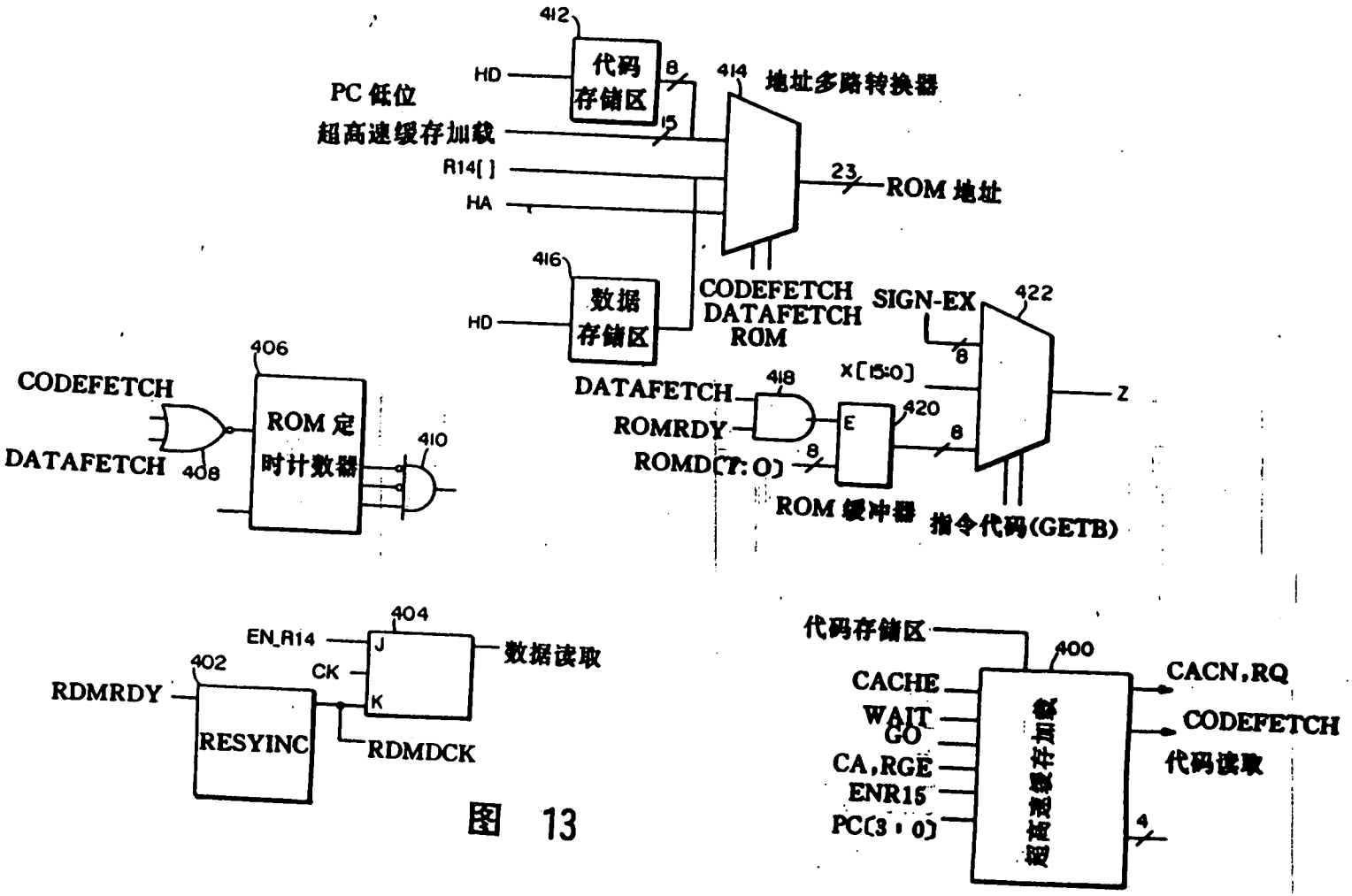
412
代码存储区
HD
8
414
地址多路转换器
PC 低位
超高速缓存加载
15
R14[ ]
HA
23
ROM 地址
416
数据存储区
HD
CODEFETCH
DATAFETCH
ROM
SIGN-EX
422
8
X[15:0]
DATAFETCH
418
ROMRDY
E
420
8
ROMD[7:0]
8
Z
ROM 缓冲器
指令代码(GETB)
406
CODEFETCH
ROM 定时计数器
DATAFETCH
408
410
404
EN_R14
J
数据读取
402
CK
RDMRDY
K
RESYINC
RDMDCK
代码存储区
400
CACHE
CACN,RQ
WAIT
GO
CODEFETCH
CA,RGE
代码读取
ENR15
PC[3:0]
4
超高速缓存加载

图 13

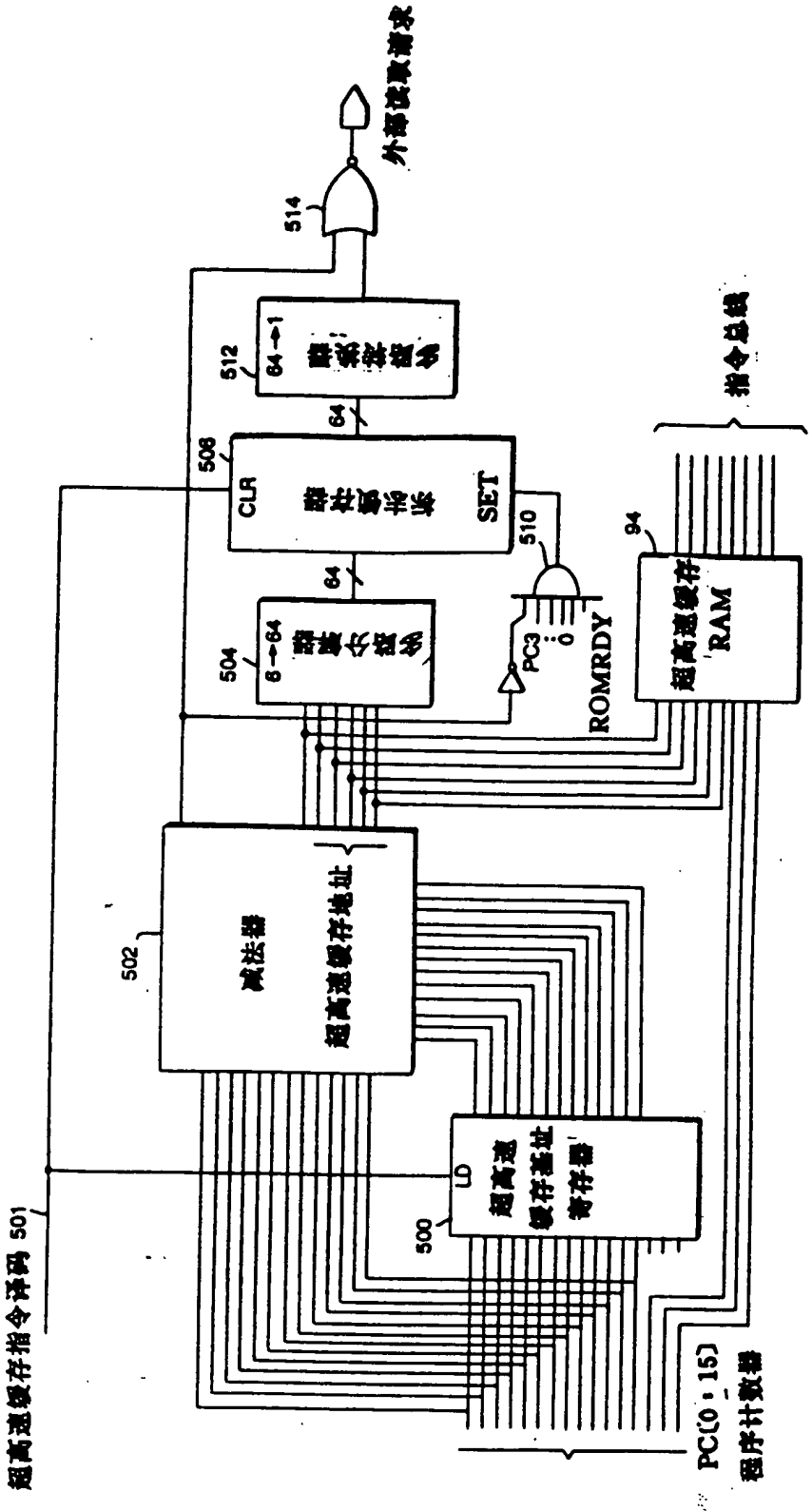
外部读取请求
514
512
64—1
多路转换器
64
506
CLR
地址锁存器
SET
510
64
504
8—64
多路分配器
PC3
0
ROMRDY
94
超高速缓存
RAM
指令总线
502
减法器
超高速缓存地址
超高速
缓存地址
寄存器
LD
500
超高速缓存指令译码 501
PC(0:15)
程序计数器

图 14

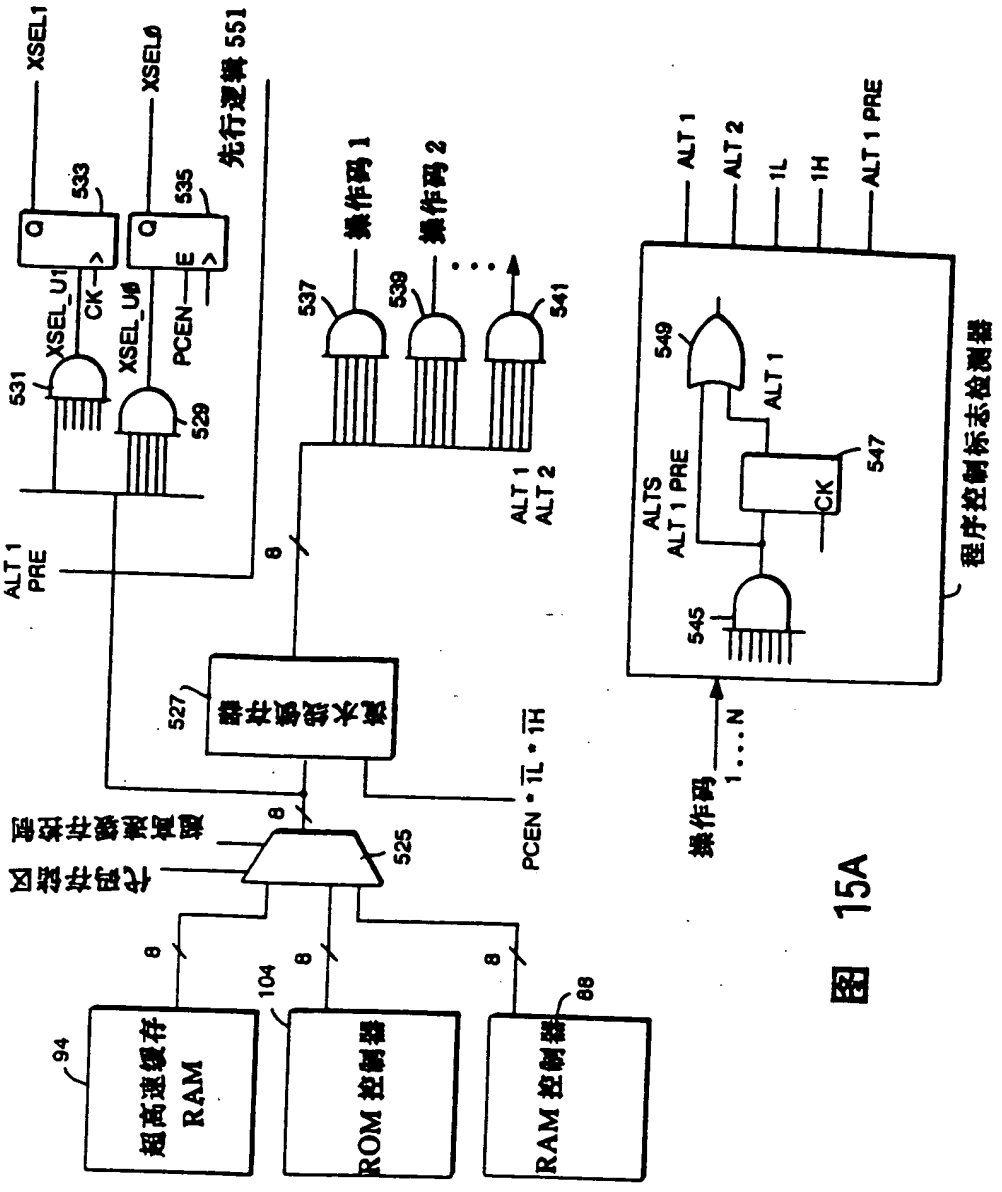
XSEL1
XSEL0
先行逻辑 551
533
535
Q
D
E
CK
XSEL_U1
XSEL_U0
PCEN
531
529
537
539
541
操作码 1
操作码 2
ALT 1
ALT 2
ALT 1
PRE
8
527
525
PCEN • $\overline{1L}$ • $\overline{1H}$
8
8
8
8
104
88
94
超高速缓存
RAM
ROM 控制器
RAM 控制器
ALT 1
ALT 2
1L
1H
ALT 1 PRE
549
ALT 1
547
CK
ALTS
ALT 1 PRE
545
程序控制标志检测器
操作码
1 . . . N

图 15A

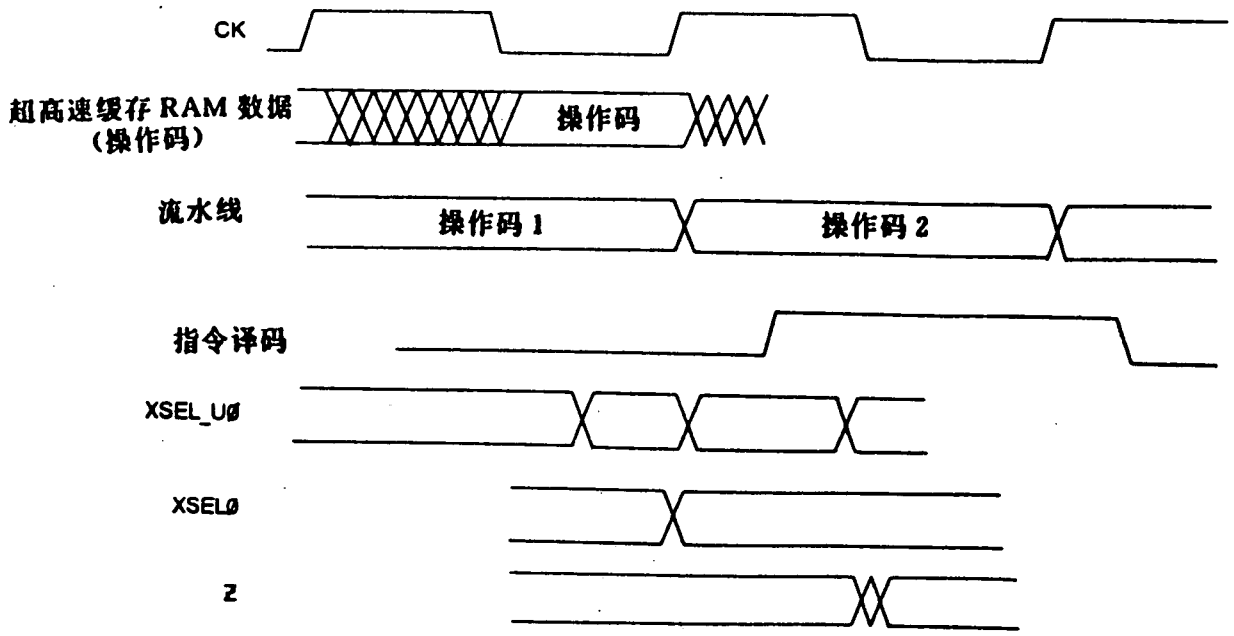

图 15B

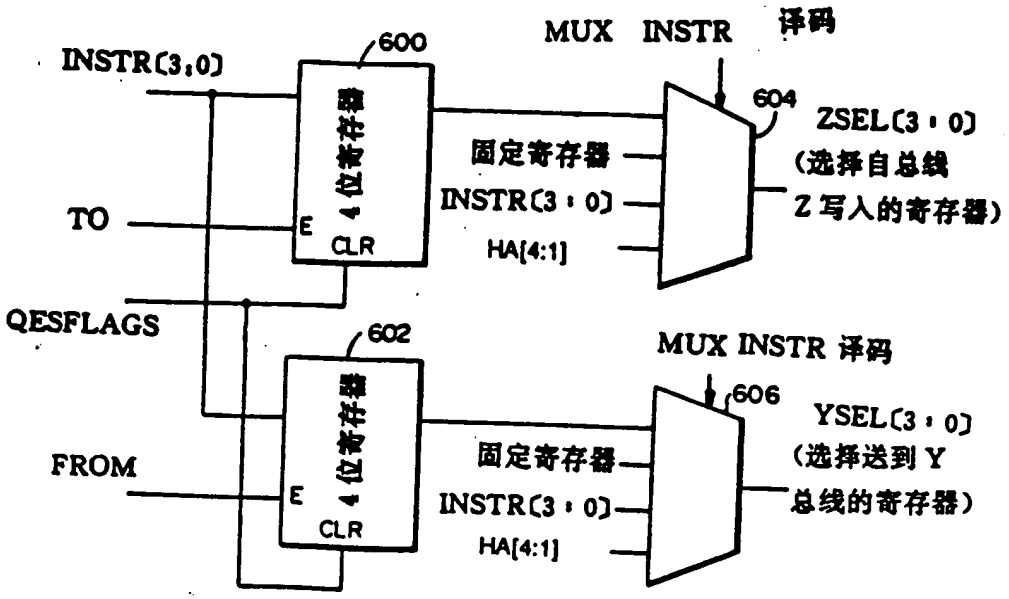
INSTR[3:0]
600
MUX INSTR 译码
604
ZSEL[3:0]
(选择自总线
Z 写入的寄存器)
固定寄存器
INSTR[3:0]
HA[4:1]
TO
E
4 位寄存器
CLR
QESFLAGS
602
MUX INSTR 译码
606
YSEL[3:0]
(选择送到 Y
总线的寄存器)
FROM
固定寄存器
INSTR[3:0]
HA[4:1]

图 16

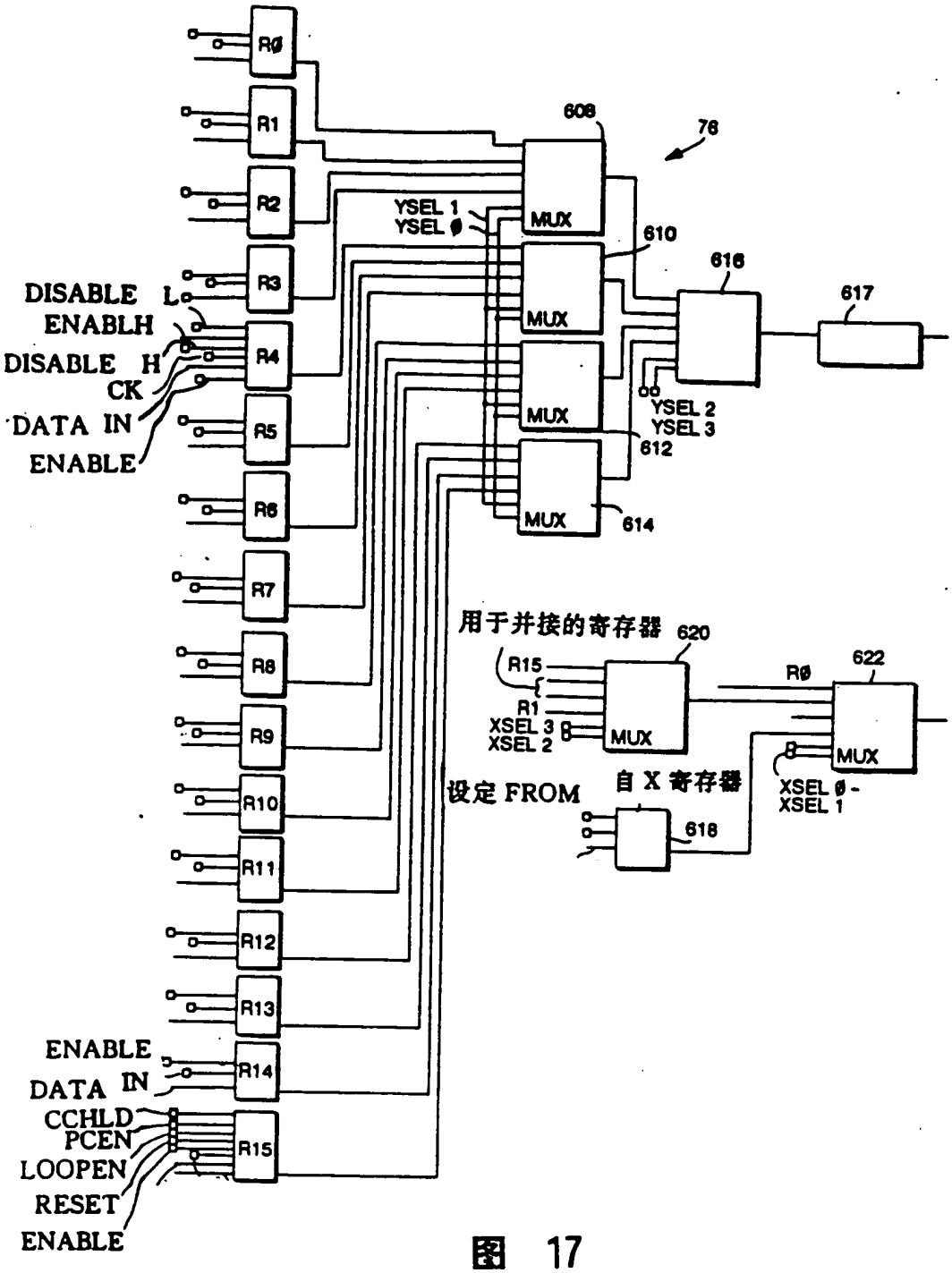
R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
608
76
YSEL 1
YSEL 0
MUX
610
616
617
YSEL 2
YSEL 3
612
614
DISABLE L
ENABLH
DISABLE H
CK
DATA IN
ENABLE
用于并接的寄存器
620
622
R15
R1
XSEL 3
XSEL 2
R0
XSEL 0 -
XSEL 1
设定 FROM
自 X 寄存器
618
ENABLE
DATA IN
CCHLD
PCEN
LOOPEN
RESET
ENABLE

图 17

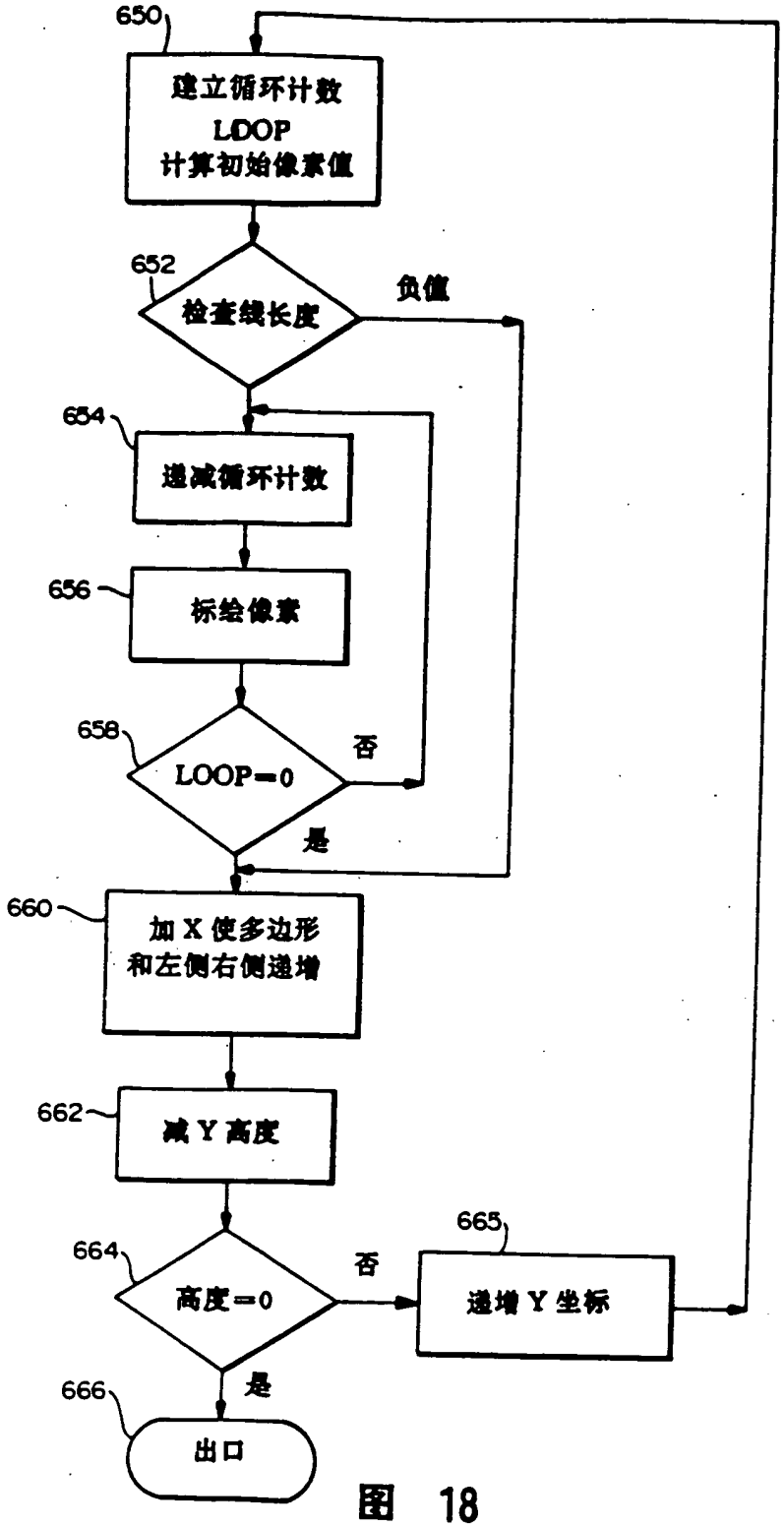
650
建立循环计数
LOOP
计算初始像素值
652
检查线长度
负值
654
递减循环计数
656
标绘像素
658
LOOP=0
否
是
660
加 X 使多边形
和左侧右侧递增
662
减 Y 高度
664
高度=0
否
665
递增 Y 坐标
是
666
出口

图 18

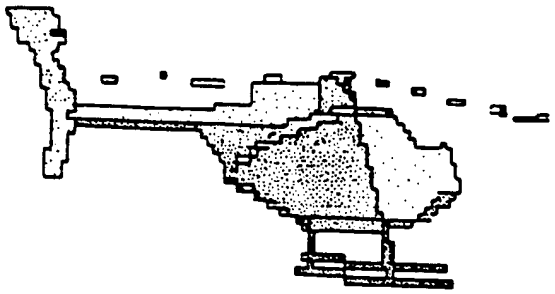

图 19

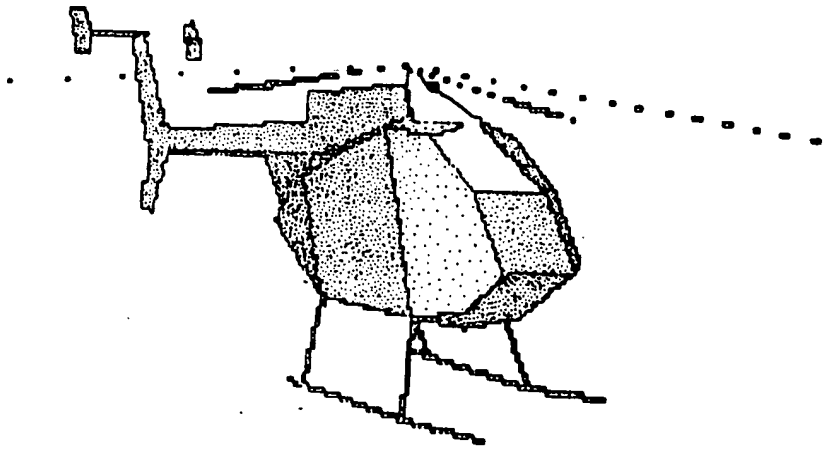

图 20

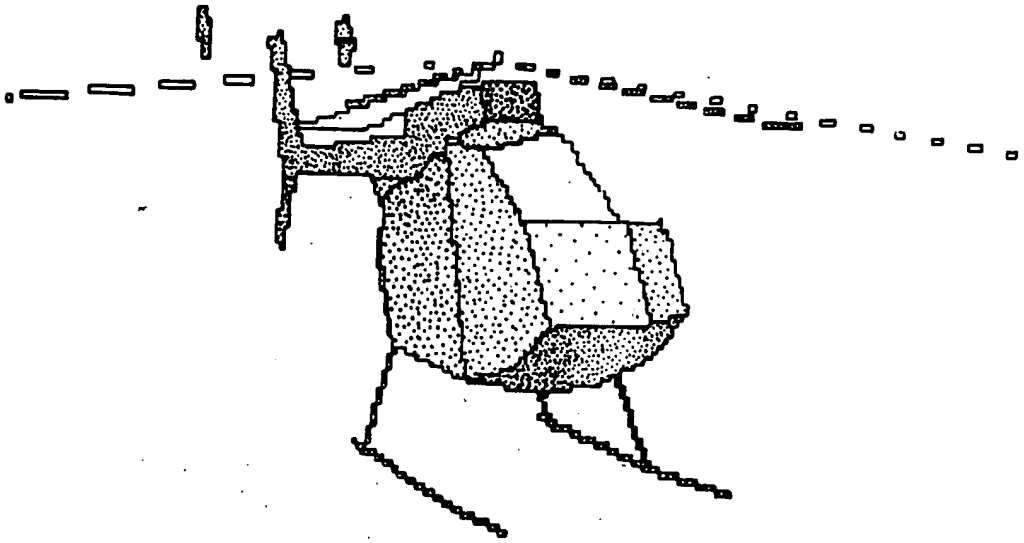

图 21